

BRÜCKMANN ENGLISCH GERITS

CPC 464 INTERN

MIT KOMMENTIERTEM ROM-LISTING

EIN DATA BECKER BUCH



BRÜCKMANN ENGLISCH GERITS

CPC 464

INTERN

MIT KOMMENTIERTEM ROM-LISTING

EIN DATA BECKER BUCH

ISBN 3-89011-080-0

Merowingerstr. 30
4000 Düsseldorf

Wichtiger Hinweis!



INHALT

Kapitel Seite

Kapitel Seite

Einleitung

Als wir im Herbst 1984 den ersten CPC 464 erhielten, waren wir zuerst skeptisch. 'Einer von vielen', so dachten wir, bevor wir die Leistungsfähigkeit des Rechners kannten.

Wie gründlich wir unsere Meinung schon nach kurzer Zeit revidierten, können Sie nicht nur am Umfang, sondern auch am Inhalt des vorliegenden Buchs ablesen.

Der CPC 464 ist ein fantastisches Gerät mit derzeit konkurrenzlosem Preis–Leistungsverhältnis. In der Klasse der Geräte unter DM 1000.– stellt der CPC eine neue Dimension dar. Ausschlaggebend hierfür sind mehrere Punkte. Zunächst besticht die Vollständigkeit des Systems. Kein Streit um Dallas oder Sportschau trüben dank mitgeliefertem Farb– oder Grünmonitor das Familienleben, die lästigen und immer nur als Fußangeln nützlichen Verbindungskabel gehören der Vergangenheit an, der eingebaute Recorder verbannt die ewige Fummelei mit der richtigen Lautstärke in die frühe Computersteinzeit und die kostspieligen Speichererweiterungen und Interface–Karten kann man jetzt auch getrost vergessen. Es ist einfach alles da, um sofort loszulegen.

Und wie man loslegen kann. Das LOCOMOTIVE–Basic gehört unbestreitbar zum besten, was man für Geld und gute Worte bekommen kann. Besonderer Knackpunkt ist die sehr flexible und vielseitig einsetzbare Programmierung von Interrupts, die dieses Basic parat hat.
Die excellente Grafik und die Möglichkeit der Darstellung von 80 Zeichen auf dem Bildschirm ohne zusätzliche Module und Unkosten ist bislang unübertoffen. Andere Rechner unterhalb der magischen DM 1000.–Grenze haben oft schon immense Probleme, 40 Zeichen pro Zeile lesbar und flimmerfrei auf den Bildschirm zu bekommen.
Die Grafik–Auflösung von 640 x 200 Punkten ist in dieser Preisklasse genau so einmalig. Vergleichbare Leistungen bietet z.B. der IBM–PC, das allerdings nur beim mindestens fünf– bis achtfachen des Preises eines CPC 464.
Auch die Soundmöglichkeiten des CPC sind beeindruckend. Zwar ist die Erzeugung eines Stradivari–Klangs selbst bei geschicktester Programmierung nicht erzielbar, aber Sie haben sich ja auch für einen leistungsfähigen Computer und nicht für eine Geige entschieden.

Was die Geschwindigkeit angeht, so muß sich der CPC nicht verstecken. Der eingebaute Z80–Prozessor wird mit einer Taktfrequenz von 4 MHz betrieben und hat einen sehr mächtigen Befehlsvorrat. Dieser Befehlsvorrat wurde von den Entwicklern stark 'ausgereizt'. Das Ergebnis ist ein wirklich flotter Basic–Interpreter, der seinesgleichen sucht.

Aber über kurz oder lang (sicher eher kurz) kommt bei fast allen Compu-

terbesitzern der Wunsch nach mehr Information, mehr Wissen über den Computer, den man besitzt. Das wirklich lobenswert gute Bedienungs-—Handbuch zum CPC allein reicht nicht aus. Besonders gilt dies dann, nachdem das Basic etwas an Reiz verloren hat, und es gilt, die durch Basic gesteckten Grenzen in Richtung Maschinensprache zu überschreiten. Dann werden Informationen nötig, die weit über das hinausgehen, was ein Bedienungshandbuch geben kann.

Diese Informationen wollen wir Ihnen mit dem vorliegenden Buch zur Hand geben. Alles, was wir in langer Nacht— und Tagarbeit (die Reihenfolge stimmt) über den CPC herausbekommen haben, finden Sie in diesem Buch.

Sie finden hier eine detaillierte Beschreibung der Hardware mit Schaltplan, ein sehr vollständig dokumentiertes Listing des Betriebssystems und des Basic, wichtige Adressen im Ram, aber auch Basic—Befehle, die im Handbuch nicht beschrieben sind. Weiter sind Tricks im Umgang mit der Cassette und dem Drucker enthalten, sowie die Programmierung der Grafik in Maschinensprache.

Wir hoffen, daß Ihnen die gebotene Information nützlich ist, und Sie Ihren CPC noch besser verstehen.

Ihre Autoren

1 DIE HARDWARE

1.1 Das sollten Sie von Ihrem Gerät wissen.

Sie haben noch nicht zum Schraubendreher gegriffen, um das Innenleben dieses 'Wunderkastens' zu betrachten? Macht nichts, wir haben Ihnen die Arbeit des Aufschraubens abgenommen und das Ergebnis für Sie photografiert. Bild 1.1.0.1 zeigt, wie es im Innern des Gerätes aussieht.

Ganze 25 ICs sind auf einer recht großen Leiterplatte großzügig verteilt. Die Leistungsfähigkeit des CPC ist also nicht durch besonders aufwendige Elektronik erreicht, eher ist es die Software, durch die der Rechner so außergewöhnlich wird. Da ist dann auch der erstaunlich niedrige Preis für das Gesamtsystem verständlich. Die paar Bauteile im CPC kosten ja nicht viel.

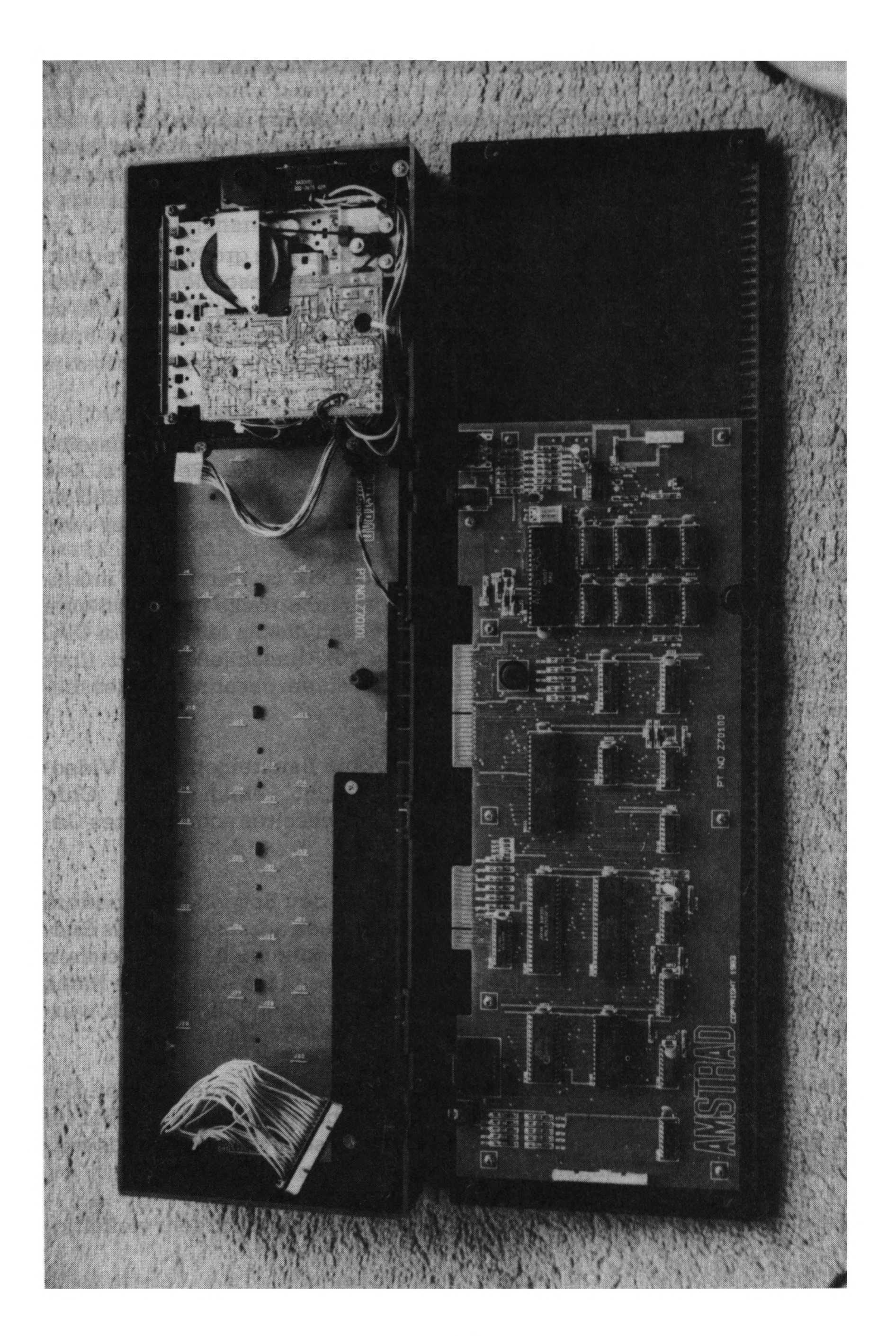
Allein 9 ICs stellen den im CPC verfügbaren Speicher dar. Acht Bausteine vom Typ 4164 sind die Rams, der Arbeitsspeicher des Rechners. Das neunte Speicher–IC ist ein Rom mit 32K–Byte. Nun hat aber der Z80 als 8–Bit–Prozessor nur einen Adressbereich von 64K–Byte und der wird durch die Ram–Bausteine vollständig ausgefüllt.
Die scheinbar unmögliche Adressierung von 96K ist durch sehr intelligente als Bank–Switching bekannte Programmiertricks und Hardware gelungen. Aber damit nicht genug. Theoretisch lassen sich an den CPC maximal 252 externe Roms mit jeweils 16K anschließen und über Bank–Switching adressieren. Das entspricht einem gesamten Adressbereich von etwa 4 Megabyte!

Außerdem enthält der CPC an hochintegrierten Bausteinen einen Video–Controller HD 6845, einen Parallelport 8255, einen Sound Chip AY–3–8912 und ein speziell für den CPC entwickeltes sogenanntes Gate Array.

Der Video–Controller hat die Aufgabe, alle für den Betrieb des Monitors benötigten Signale zur Verfügung zu stellen. Auch adressiert er das Bildschirmram, den Speicherbereich, in dem die darzustellenden Zeichen und die Grafik abgelegt werden. Zusätzlich erzeugt er den für die Rams nötigen Refresh, ohne den diese ihre Information schnell verlieren würden.

Die Aufgabe des Sound Chip geht schon aus seinem Namen hervor. Die von den Konstrukteuren getroffene Wahl ist sehr gut. Der AY–3–8912 ist in vielen Computern eingesetzt worden, weil er sehr vielseitige und weitreichende Beeinflussungsmöglichkeiten des Sounds bietet.

Der 8255 ist das 'Arbeitstier' im CPC. Seine Aufgaben sind sehr vielfältig.


AMSTRAD
PT NO Z70100

Das beginnt bei der Kontrolle der Tastatur, geht über die Ansteuerung des Sound Chip weiter zur Steuerung des Recorders, bestimmt verschiedene Möglichkeiten des CPC u.s.w.

Besonders interessant ist das Gate Array. Dieser Baustein steuert so viele Dinge im CPC, daß man ihm fast den Rang eines Hilfsprozessors zusprechen könnte. So werden viele den Bildschirm betreffende Aufgaben von ihm übernommen. Dazu gehört unter anderem die Darstellung der verschiedenen Farben und die unterschiedlichen Zeilenformate. Weiter werden alle benötigten Taktsignale im Gate Array erzeugt. Der Interrupt, der 300 mal in der Sekunde den normalen Programmablauf unterbricht, wird, wie auch die Signale für die Verwaltung der 96 K Speicher im CPC, vom Gate Array generiert.

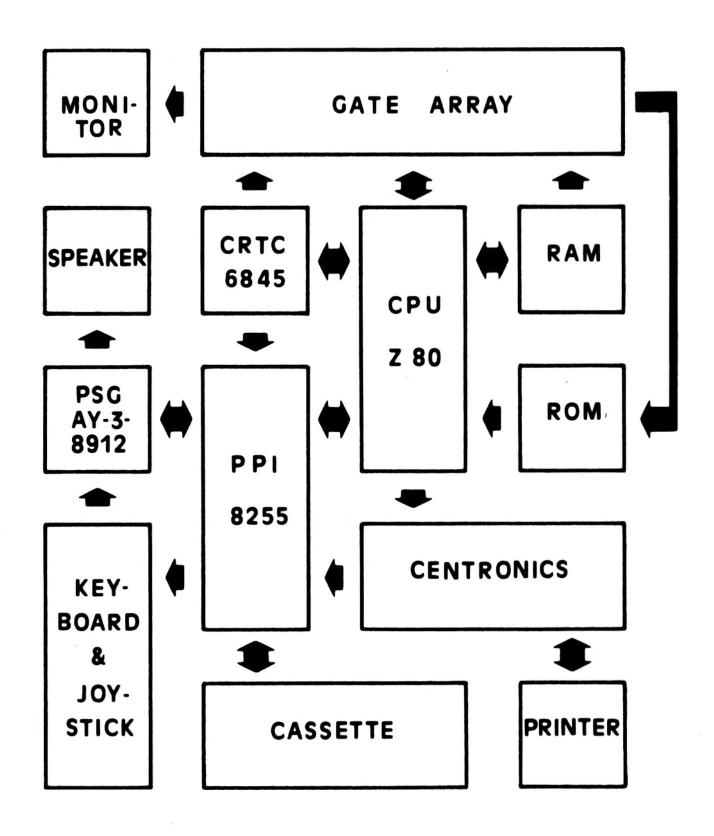
Wie die einzelnen Komponenten zusammenspielen, zeigt das Blockschaltbild 1.1.0.2.

1.1.1 Die Speicheraufteilung

Noch vor 5 Jahren galten Computer mit 16 K Ram als recht gut bestückt. Spätestens seit dem Erscheinen des C64 von Commodore haben sich die Speichergrenzen deutlich verschoben. Ein Computerhersteller kann sich nur dann ausreichend Marktchancen ausrechnen, wenn auf seinem Gerät wenigstens die 'magische 64' erscheint.
Auch der CPC ist mit einem Rambereich von 64 K = 65536 Speicherplätzen ausgerüstet. Zusätzlich sind noch 32 K ROM eingebaut.

Nun ist es nicht besonders schwer, 64 K Speicher in einem Computer unterzubringen, da die gebräuchlichen 8-Bit-Prozessoren diesen Speicherbereich alle adressieren können. Auch der Z80 des CPC kann 64 K Speicher ohne Tricks adressieren. Das reicht aber gerade für den Ram-Speicher des CPC, dann geht schon nichts mehr.
Will man aber mehr Speicher mit diesen Prozessoren adressieren, so hat man mehrere Möglichkeiten, dies zu tun. Neben solch hardwareorientierten Möglichkeiten wie dem Einsatz spezieller Expander zum Erreichen eines Adressbusses mit 19 oder 20 Adressleitungen gibt es auch mehr softwaremäßige Mittel. Einer dieser Wege ist das 'Bank-Switching'-Verfahren.
Bei diesem Verfahren werden in sich überlappenden Speicherbereichen (den sogenannten Banks) wahlweise Ram oder Rom angesprochen. Im Gegensatz zu den rein hardwaremäßigen Lösungen benötigt man für dieses Verfahren zwar eine Software, die das Miteinander von Rom und Ram auf denselben Adressen organisiert. Die aber ist den Entwicklern sehr elegant gelungen.

Im CPC ergibt sich nun das folgende Bild. Durchgängig adressiert werden 64 K Ram. 'Parallel' dazu liegt in den untersten 16 K (&0000 bis



1.1.0.2 Blockschaltbild des CPC

&3FFF) eine Hälfte des Rom, die zweite Hälfte liegt in den oberen 16 K (&C000 bis &FFFF).
Die unteren 16 K Rom beinhalten das Betriebssystem und ein Paket von Routinen für die Arithmetik. Im Betriebssystem finden sich alle Routinen, die der CPC benötigt, um z.B. ein Zeichen von der Tastatur zu lesen, ein Zeichen oder einen Grafik-Punkt auf den Bildschirm zu bringen, aber auch der Recorder und die Drucker-Schnittstelle sowie der Sound wird über das Betriebssystem bedient.
In den oberen 16 K befindet sich der Basicinterpreter. Diese 16 K haben noch eine besondere Bewandtnis. In diesen Bereich können bis zu 252 weitere Roms geschaltet werden. So sind z.B. alle für den Betrieb der Floppy benötigten Routinen in einem Rom untergebracht, das sich den Platz mit dem Basic 'teilt'.

Grafisch kann man die Speicheraufteilung wie im Bild 1.1.1.1 darstellen.

1.1.2 Befehlserweiterung via RST

Allerdings stellt sich bei dieser Art der Ansteuerung die Frage, wie der Zugriff auf die Roms oder das darunter liegende Ram geschehen kann. Für diese Aufgaben, die sonst einen nicht unerheblichen Programmieraufwand bedeuten würden, haben sich die Programmierer des Betriebssystems einen schönen Trick einfallen lassen. Durch spezielle Programme und der geschickten Ausnutzung der RESTART-Befehle des Z80 ergibt sich für die Restarts RST1 bis RST5 quasi eine Erweiterung des Befehlssatzes des Z80. Diese RSTs lassen sich wie übliche JPs oder CALLs einsetzen. Bei einigen RSTs wird allerdings eine 3-Byte-Adresse verlangt. Im zusätzlichen dritten Byte wird dann bestimmt, in welches ROM der JP oder CALL gehen soll.

LOW JUMP RST 1

Dieser Befehl dient zum Aufruf einer Routine im Betriebssystem oder im darunterliegenden RAM. Direkt hinter dem RST-Befehl muß die Adresse der aufzurufenden Routine stehen. Da für den Bereich von 0 bis &3FFF 14 Adreßbits ausreichen, benutzt man die oberen beiden Bits für die Auswahl von ROM oder RAM:

Bit 14 = 0	Betriebssystem ausgewählt
Bit 14 = 1	RAM ausgewählt
Bit 15 = 0	BASIC-ROM ausgewählt
Bit 15 = 1	RAM ausgewählt

Ein Aufruf der Betriebssystemroutine &1410 könnte dann so aussehen:

```
RST 1
DW &1410 + &8000
```

FFFF

16 K
(VIDEO)
RAM

16 K
BASIC
ROM

16 K
max. 252
ext. ROMS

C000

16 K
RAM

8000

16 K
RAM

4000

16 K
RAM

16 K
Betriebs-
system
ROM

1.1.1.1 Speicheraufteilung im CPC

Durch das gesetzte Bit 15 ist im Bereich von &C000 bis &FFFF RAM selektiert, während durch das gelöschte Bit 14 das Betriebssystem angesprochen wird.

Der Kode an der Adresse 8 besteht lediglich aus einem Sprung zu &B982.

SIDE CALL RST 2

Dieser Restart-Befehl dient zum Aufruf einer Routine in einem Expansion-ROM. Der Befehl wird dann benutzt, wenn ein Programm, das als ROM-Modul vorliegt, mehr als 16 KByte benötigt und nicht mehr in einem Expansion-ROM Platz hat. Dann kann mit Hilfe des SIDE CALL eine Routine im zweiten, dritten oder vierten zugehörigen ROM aufgerufen werden, ohne daß man die absolute Nummer des jeweiligen ROMs kennen muß. Nach dem RST 2-Befehl muß die Adresse der Routine - &C000, d.h. also die relative Adresse bezogen auf den Start des ROMs, stehen. Die obersten beiden Bits werden zur Auswahl der vier verschiedenen ROMs benutzt.

An Adresse &0010 steht ein Sprung zu &BA16.

FAR CALL RST 3

Mit Hilfe dieses RST-Befehls können Sie eine Routine irgendwo im ROM oder RAM aufrufen. Dazu muß hinter dem RST 3-Befehl die 2-Byte-Adresse eines Parameterblocks stehen, der aus drei Bytes besteht. Diese ersten beiden Bytes enthalten die Adresse der Routine, die aufgerufen werden soll, und das dritte Byte muß den gewünschten ROM/RAM-Status enthalten. Dabei wird durch die Werte von 0 bis 251 das entsprechende Zusatzrom angesprochen. Die verbleibenden vier Werte haben folgende Funktion:

Wert	&0000−&3FFF	&C000−&FFFF
252	Betriebssystem	BASIC
253	Betriebssystem	RAM
254	RAM	BASIC
255	RAM	RAM

An Adresse &0018 steht ein Sprung nach &B9BF.

RAM LAM RST 4

Mit Hilfe dieses RST-Befehls können Sie von einem Maschinenprogramm den Inhalt des RAMs lesen, unabhängig vom jeweils gewählten ROM-Zustand. Der RST 4-Befehl ersetzt dabei den Befehl.

```
LD   A,(HL)
```

HL muß dazu also die Adresse der zu lesenden Speicherzelle enthalten. An Adresse &0020 steht ein Sprung zu &BACB.

FIRM JUMP RST 5

Mittels dieses RST-Befehls kann man zu einer Routine im Betriebssystem springen. Die Adresse muß dabei unmittelbar auf den RST 5-Befehl folgen. Das Betriebssystem-ROM wird enabled, bevor die Routine angesprungen wird und wird bei der Rückkehr wieder disabled. An Adresse &0028 steht ein Sprung zu &BA2E.

1.2 Der Prozessor Z80

In den frühen 70er Jahren begann der Siegeszug der Microprozessoren. Die Firma INTEL konnte mit dem Prozessor 8080 einen bedeutenden Marktanteil erreichen, da zum Zeitpunkt der Markteinführung in dieser Klasse praktisch keine Konkurrenz vorhanden war. Dies macht sich allerdings auch bemerkbar, wenn die Leistungsdaten des Prozessors genauer untersucht werden. So benötigt der 8080 noch drei verschiedene Betriebsspannungen und zwei weitere ICs zur Steuersignalerzeugung und Taktgenerierung.

In den Jahren 74/75 wurde von der Firma ZILOG der Z80 entwickelt. Anstatt aber einen von Grund auf neuen Prozessor zu entwickeln, hielt man sich an das so gut angekommene Konzept des 8080. Aus diesem Grunde ist der Z80 zum 8080 aufwärts–kompatibel, d.h. alle für einen 8080 geschriebenen Programme laufen auch auf einem Z80–Prozessor.

Allerdings wurden alle mittlerweile beim 8080 als ungünstig erkannten Eigenschaften beseitigt und der Befehlssatz wurde stark erweitert. Auch benötigt der Z80 nur eine Betriebsspannung von +5Volt und aufwendige externe ICs zur Steuersignalerzeugung sind überflüssig.

Doch betrachten wir im Telegrammstil die Leistungsdaten des Prozessors, bevor wir auf seine Eigenschaften etwas konkreter eingehen.

8–Bit–Prozessor in NMOS–Technologie
16–Bit–Adressbus
Einfache Stromversorgung 5 Volt
Einfacher Takt
TTL–Kompatibel
Wahlweise 2.5, 4, 6 oder sogar 8 MHz Taktfrequenz
Softwarekompatibel mit 8080
Doppelter Registersatz, zusätzlich zwei Indexregister
Nicht maskierbarer Interrupteingang
Maskierbarer Interrupteingang mit drei Betriebsarten
Selbstätiger Refresh von dynamischen Rams
8080–Peripherie–ICs direkt anschließbar

Diese Leistungsdaten und die große Menge an fertiger Software haben den Z80 zu einem der erfolgreichsten 8–Bit–Prozessoren werden lassen. Im Bereich der Home– und Personalcomputer hat nur ein weiterer Prozessor, der 6502, eine vergleichbare Verbreitung gefunden.

1.2.1 Die Anschlüsse des Z80

Nach diesem kurzen Überblick über die Leistungsmerkmale wollen wir zunächst die Belegung der 40 Pins des Z80 betrachten.

A 11 (1) — A 10
A 12 — A 9
A 13 — A 8
A 14 — A 7
A 15 — A 6
Ø — A 5
D 4 — A 4
D 3 — A 3
D 5 — A 2
D 6 — A 1
+5V — A 0
D 2 — GND
D 7 — RFSH*
D 0 — M1*
D 1 — RESET*
INT* — BUSRQ*
NMI* — WAIT*
HALT* — BUSAK*
MREQ* — WR*
IORQ* — RD*

1.2.1.1 Pinout des Z80

Die Anschlüsse des Z80 lassen sich in die vier Gruppen Datenbus, Adressbus, Steuerbus und Versorgungsleitungen zusammenfassen.

Adressbus

A0 – A15 : **Adresslines**, über diese Anschlüsse wird eine Speicherzelle im Adressbereich angewählt. Der Adressbereich umfasst 65536 Speicherplätze. Bei der Behandlung der I/O–Befehle werden die unteren 8 Adressbits benutzt, um die entsprechende I/O–Adresse auszugeben. Somit sind 256 verschiedene Ports möglich. Mit gewissen Einschränkungen im Befehlssatz können aber sogar 65536 Ports adressiert werden. Dann werden alle 16 Adressleitungen zur Bildung der Portadresse herangezogen. Auf diesen Spezialfall werden wir später zurückkommen.

Datenbus

D0 – D7 : **Datalines**, über diese bidirektionalen Leitungen gelangen die Daten von und zum Prozessor. Sie stellen die Verbindung zwischen Prozessor und der durch den Adressbus ausgewählten Speicherzelle oder auch Portadresse her.

Steuerbus

M1* : **Machine Cycle One**, dieses Steuersignal zeigt an, daß der Prozessor den Operationscode vom Datenbus liest. Der Stern deutet übrigens bei diesem und den folgenden Signalen an, daß es sich hierbei um lowaktive Signale handelt.

MREQ* : **Memory REQuest***, dieses Ausgangssignal zeigt durch ein Low an, daß der Prozessor einen Schreib– oder Lesezugriff auf eine Speicheradresse vornimmt und die Adresse auf dem Adressbus gültig ist.

IORQ* : **Input/Output ReQuest***, ein Low dieses Ausgangs zeigt an, daß der Prozessor einen Schreib– oder Lesezugriff auf eine Portadresse vornimmt und die Portadresse auf dem Adressbus gültig ist.

RD* : **ReaD***, dieses Ausgangssignal ist Low, wenn der Prozessor Daten aus einer Speicherzelle oder Portadresse lesen will. Durch Verknüpfung mit MREQ* und IORQ* kann zwischen Lesen aus Speicher und Port unterschieden werden.

WR* : **WRite***, dieses Signal des Z80 wird Low, wenn bei Schreibzugriffen des Z80 auf Speicher– oder Portadressen die Daten auf dem Datenbus gültig sind. Auch hier kann wieder durch Verknüpfen des WR* mit MREQ* und IORQ* unterschieden werden, ob Daten in den Speicher oder eine Portadresse geschrieben werden.

RESET* : Wird dieser Eingang auf Low gelegt, dann wird der Programmzähler mit dem Wert &0000 geladen, Interrupts werden gesperrt und der Interruptmodus 0 wird eingeschaltet. Sobald der Eingang wieder High wird, beginnt der Prozessor das Programm ab der Adresse &0000.

NMI* : **Non Maskable Interrupt***, durch eine High–Low–Flanke an diesem Eingang wird der Prozessor immer im laufenden Programm unterbrochen. Der Programmzähler wird mit den in den Adresse &0066 und &0067 gespeicherten Werten geladen und an dieser Stelle wird das Programm fortgesetzt.

IRQ* : **Interrupt ReQuest***, durch ein Low an diesem Eingang kann der Prozessor im laufenden Programm unterbrochen werden, wenn diese Art des Interrupt per Befehl freigegeben ist. Die Auswirkungen unterscheiden sich je nach gewähltem Interruptmodus und werden später besprochen. IRQ* stellt im Gegensatz zu NMI* ein statisches Signal dar und muß bis zum Erkennen der Interruptanforderung anliegen.

WAIT* : Mit Hilfe dieses Signals kann der Lese– oder Schreibzugriff des Z80 an langsamere Speicher oder spezielle Bedingungen des Systems angepasst werden.

BUSRQ* : **BUSReQuest***, wird dieser Eingang Low, dann werden nach der Abarbeitung des laufenden Befehls Adress– und Datenleitungen sowie alle Ausgangssteuerleitungen hochohmig und das BUSAK*–Signal wird Low. Jetzt könnte ein zweiter Prozessor den Zugriff auf den Speicher und die Peripheriebausteine übernehmen, hauptsächlich wird dieses Signal jedoch für DMA benutzt (DMA=Direkt Memory Access, sehr schneller Datentransfer bei Umgehung des Prozessors).

BUSAK* : **BUSAKnowledge***, stellt das mit BUSRQ* korrespondierende Ausgangssignal dar. Ein Low zeigt dem DMA–Controller oder zweiten Prozessor an, daß alle Steuer– und

Bussignale hochohmig sind und ein Zugriff jetzt erfolgen kann.

HALT* : Dieser Ausgang wird Low, nachdem der Prozessor den Maschinensprache-Befehl HALT ausgeführt hat. Nach diesem Befehl 'tut' der Prozessor nichts mehr, er führt NOPs aus, um den Refresh sicherzustellen. Nur ein Interrupt kann ihn wieder 'wecken'.

RFSH* : **ReFreSH***, dieses Ausgangssignal zeigt an, daß auf den unteren sieben Adressleitungen eine gültige Refresh-Adresse liegt. Da der Prozessor nur zu bestimmten Zeiten den Adress- und Datenbus benötigt, kann in der verbleibenden Zeit der Adressbus zum Auffrischen dynamischer Rams verwendet werden, ohne daß aufwendige Elektronik oder spezielle Auffrisch-Routinen benötigt werden.

Takt und Stromversorgung

Ø : Der Eingang Phi liefert den Takt für den Prozessor. Da der Z80 ein statisches IC ist, kann der Takt von 0 Hertz bis zur angegebenen Maximalfrequenz betragen. Allerdings werden an die Form des Taktsignals bestimmte Anforderungen gestellt. Laut Datenblatt darf die maximale Lowzeit dieses Signals 2 μs (Microsekunden) betragen. Dieser Wert ist allerdings mehr von akademischem Interesse, da man ja bemüht sein wird, den Prozessor mit möglichst hoher Taktfrequenz zu versorgen, um ein schnelles Abarbeiten des Programms zu erhalten.

GND : Masseanschluß des Prozessors.

Vcc : Über diesen Anschluß bekommt der Z80 seinen Saft, sprich +5 Volt Gleichspannung und ca. 150 bis 200 Milliampere.

1.2.2 Der Register–Aufbau des Z80

Wie schon zu Beginn erwähnt, ist der Z80 so konstruiert worden, daß Programme des 8080 ohne weiteres übernommen werden können. Allerdings ist die Anzahl der Register des Z80 deutlich höher.

Aber was ist eigentlich ein Register?

Nun, ein Register ist nichts anderes als ein Schreib/Lese–Speicher auf dem Prozessorchip. Jeder Prozessor muß eine Mindestzahl von Registern aufweisen. In diesen Speicherzellen werden Daten gespeichert und die Ergebnisse von arithmetischen und logischen Befehlen abgelegt. Andere Register haben spezielle Aufgaben, wie die Verwaltung des Stack oder werden als Programmzähler verwendet.

Da Operationen wie ein Transfer von Daten zwischen zwei Registern oder die Addition zweier Registerinhalte nicht über den Datenbus abgewickelt werden, kann eine solche Operation sehr viel schneller durchgeführt werden, als wenn die benötigten Werte aus externen Speicherplätzen geholt werden müssen.

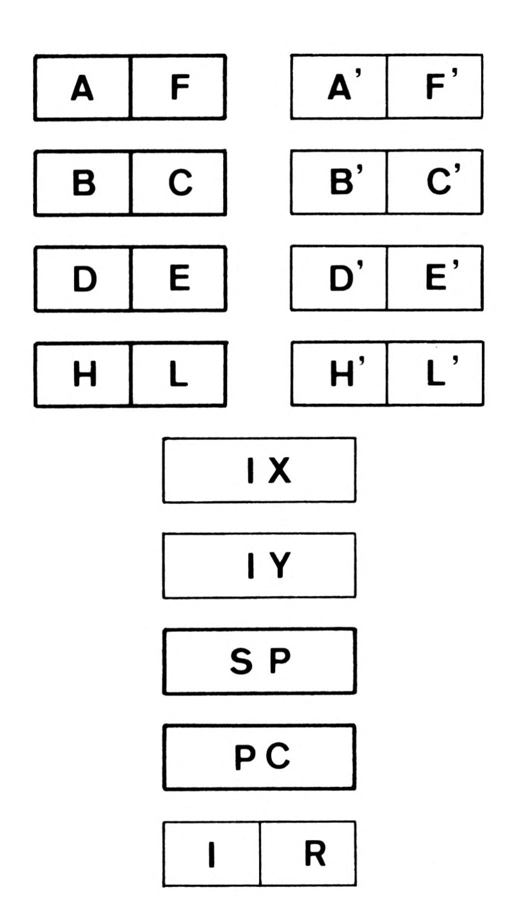
Als grobe Regel kann man sagen, daß Prozessoren mit vielen Registern denen mit weniger internem Speicher bei der Bearbeitung gleicher Programme überlegen sind, da der Datentransfer innerhalb des Prozessors immer schneller ist, als ein Transfer von und zu externen Speicherplätzen.

Insgesamt verfügt der Z80 über 22 Register, 18 Register mit 8 Bit und vier 16–Bit–Register. Die Aufteilung zeigt die Grafik 1.2.2.1.

In der Grafik fallen einige Register durch ihre stärkere Umrahmung auf. Diese Register sind auch im 8080 enthalten.
Auch ist auffällig, daß die meisten 8–Bit–Register doppelt vorhanden sind. Dies sind die Register A, F, B, C, D, E, H und L. Der Z80 stellt sie in doppelter Ausführung zur Verfügung und der Programmierer kann per Befehl zwischen den beiden Sets wählen.

Wir werden zukünftig nur von einem Registersatz sprechen. Das ist beim CPC auch insofern richtig, da ohne spezielle Tricks dem Programmierer beim CPC 464 sowieso nur ein Registersatz zur Verfügung steht. Der alternative Registersatz wird vom Betriessystem für die Interruptsteuerung benutzt. Merken Sie sich aber, daß alle Aufgaben eines Registersatzes auch vom alternativen Registersatz übernommen werden können, wenn dieser nicht für spezielle Zwecke belegt ist.

Die Register B bis L stellen allgemein verfügbare 8–Bit–Register dar, während den Registern A und F besondere Aufgaben zukommen.



1.2.2.1 Registersatz Z 80

Das A–Register wird allgemein als Akku oder Akkumulator bezeichnet. Im Akku erhält man das Ergebnis von allen arithmetischen und logischen Operationen mit 8–Bit–Format. Auch muß im Akku bei diesen Operationen ein Operand gespeichert sein. Um z.B. zwei Bytes zu addieren, ist es nötig, einen Operanden in den Akku zu speichern, der andere Operand kann in einem anderen Register oder außerhalb des Prozessors im Speicher untergebracht sein. Nach der Addition steht dann das Ergebnis im Akku.
Da bei diesen Aufgaben das Ergebnis so groß werden kann, daß es mit nur 8 Bit nicht mehr ausgedrückt werden kann (255 + 255 = 510), wird ein weiteres Bit benötigt, um das Ergebnis korrekt darzustellen. Diese Aufgabe wird vom F–Register übernommen. Das F–Register, allgemein als Flag–Register bezeichnet, ist in einzelne Bits aufgeteilt. Eines dieser Bits hat (unter anderem) die Aufgabe, einen evtl. Übertrag (engl. Carry) solcher Additionen zu bewahren. Andere Bits zeigen, ob das Ergebnis von Rechenoperationen oder Vergleichen gleich Null ist u.s.w.

Die Register B bis L können aber nicht nur einzeln angesprochen werden. Jeweils B und C, D und E sowie H und L können zu 16–Bit–Registern zusammengefasst werden. Diese Doppelregister haben dann sinnvollerweise den Namen der beiden Einzelregister, als BC, DE und HL. Doppelregister eignen sich hervorragend zum Adressieren von Tabellen und zum Transportieren und Durchsuchen von Datenblöcken.
Dem HL–Doppelregister kommt noch eine besondere Bedeutung zu. Da der Z80 mit Befehlen zur Addition und Subtraktion von 16–Bit–Werten ausgestattet ist, fungiert das HL bei solchen Anweisungen als 16–Bit–Akku.

Nur mit 16–Bit–Werten arbeiten PC, SP, IX und IY (Anm.: Spezialisten wissen, daß die Möglichkeit besteht, die Index–Register auch byte–weise zu manipulieren, wir werden IX und IY aber als reine 16–Bit–Register betrachten).

PC ist der Programm Counter oder Programm–Zähler. Der Inhalt des PC wird als Adresse für externe Speicher auf den Adressbus gelegt. Mit jedem Befehl wird der PC automatisch inkrementiert (um eins erhöht). Bei Befehlen mit mehr als einem Byte wird der PC automatisch um die benötigte Anzahl erhöht. Werden in einem Programm Sprünge notwendig, dann wird die neue Programmadresse automatisch in den PC geladen, und der Prozessor arbeitet ab dieser Adresse weiter.

SP ist der sogenannte Stackpointer. Der Stack (oder Stapel) wird benötigt, wenn von einem Programm Unterprogramme aufgerufen werden. In diesem Fall wird automatisch die Rücksprungadresse auf dem Stack abgelegt und nach Beendigung des Unterprogramms in den PC zurückgeladen.

Die beiden 16–Bit–Register IX und IY ermöglichen durch spezielle Befehle ein besonders wirkungsvolles Arbeiten mit Tabellen.

Bleiben noch die beiden Register I und R. Das I–Register oder Interrupt–Register wird in Verbindung mit der speziellen Interrupt–Betriebsart IM3 verwendet. In diesem Modus muß der den Interrupt erzeugende Baustein auf Anforderung des Prozessors einen 8–Bit–Wert liefern. Dieser Wert als Low–Byte und der Inhalt des I–Registers als High–Byte bilden die Adresse der Interrupt–Routine.
Das R– oder Refresh–Register wird in Verbindung mit dem vom Z80 automatisch durchgeführten Refresh benötigt. Nach jedem Holen eines Befehls werden die untersten sieben Bits dieses Registers automatisch incrementiert. Das achte Bit verbleibt immer, je nach Programmierung, Null oder Eins.
Sowohl I– wie R–Register werden im CPC 464 nicht verwendet. Da über den Zustand des R–Registers keine Aussage gemacht werden kann und sich der Wert ständig ändert, kann dieses Register als Zufallsgenerator benutzt werden.

1.2.3 Besonderheiten des Z80 im CPC

Die vielfältigen Möglichkeiten des Z80 lassen den Hard- und Software-Designern freie Hand bei der Konstruktion eines Computers. Diese CPU (Central Processing Unit) kann gleichermaßen effektiv in Minimalsystemen und solch leistungsstarken Geräten wie dem CPC 464 eingesetzt werden.
Gerade die Entwickler des CPC haben tief in die Trickkiste gegriffen, um mit einem Minimum an Bauteilen ein Maximum an Leistung zu erreichen. Dabei sind zwangsläufig einige Besonderheiten entstanden, deren Kenntnis für eine effektive Programmierung und Nutzung des Gerätes besonders in Maschinensprache wichtig sind. Diese Spezialitäten sollen jetzt unter die Lupe genommen werden.

Da wäre zunächst die Interruptsteuerung des CPC.
Einzige Interrupt-Quelle im CPC ist das Gate Array, dieser fantastische Baustein, der zu einem ganz wichtigen Teil die Leistungsfähigkeit des Rechners bestimmt. Alle 3,3 Millisekunden, also 300 mal in der Sekunde, wird vom Gate Array ein kurzer Impuls erzeugt und an den IRQ*-Eingang des Z80 gelegt. Der NMI*-Eingang des Prozessors ist nicht verwendet und steht eventuellen Erweiterungen am Expansion Connector zur Verfügung.
Die Frequenz des Interrupt-Signals wird aus dem H-Sync-Signal des CRTC 6845 durch einen Frequenz-Teiler gewonnen. Diese Stufe teilt den alle ca. 65 μs erscheinenden H-Sync-Impuls durch 52.
Da der Z80 im CPC im Interrupt-Modus IM1 betrieben wird, bewirkt jeder erkannte Interrupt IRQ einen RST7 oder auch CALL &0038. Der Prozessor

unterbricht sofort sein laufendes Programm, legt den derzeitigen Stand des PC auf den Stack und verzweigt zur Adresse &0038. Hier steht beim CPC ein Sprung zur Adresse &B939, der eigentlichen Interrupt-Routine. Die Stelle, an der die Unterbrechung aufgetreten ist, wird auf dem Stack vermerkt. So kann nach Beendigung der Interrupt-Routine das unterbrochene Programm fortgesetzt werden.
Da der IRQ*-Eingang des Prozessors auch am Expansion Connector anliegt, stellt sich natürlich die Frage, wie zwischen einem Interrupt vom Gate Array und einem externen Interrupt unterschieden werden kann. Hier haben die Entwickler einen speziellen Trick benutzt. Innerhalb der Interrupt-Routine ab &B939 wird für einen kurzen Moment der Interrupt wieder zugelassen. Da der vom Gate Array erzeugte Impuls nur maximal 5 μs lang ist, hat das Zulassen keine Auswirkungen, der Impuls ist lange vorbei. Externe Interruptquellen nehmen ihr Signal aber erst auf ausdrückliche Anweisung des Prozessors zurück. Liegt ein solcher externer Interrupt vor, dann wird die Interrupt-Routine also selbst unterbrochen. Dieser Fall kann aber erkannt und speziell abgehandelt werden. Dadurch sind auch externe IRQ*-Quellen möglich. Die einzige Forderung an sie ist ein ausreichend langer Impuls.

Der zweite Spezialfall, den es zu berücksichtigen gilt, ist die eingeschränkte Verwendbarkeit der Port-Befehle.
In Verbindung mit dem Signal IORQ* (Input/Output ReQuest) kann der Z80 maximal 256 verschiedene Ports analog zu Speicherplätzen adressieren. Dazu wird die Adresse des gewünschten Ports auf die unteren 8 Adressbits A0 bis A7 gelegt. Diese Ports werden vornehmlich zum Anschluß von Peripherie-Bausteinen genutzt.
Bei anderen Prozessoren, die die Möglichkeit der Port-Adressierung nicht kennen, ist der Entwickler immer darauf angewiesen, die Peripherie-Bausteine als Speicherplätze zu adressieren. Dies Verfahren nennt man Memory-Mapped und hat den Nachteil, daß der für RAM zur Verfügung stehende Adressbereich kleiner wird.
Für die Benutzung der Port-Adressierung stellt der Z80 die sehr leistungsfähige Gruppe von IN- und OUT-Befehlen zur Verfügung. Studiert man die Befehle dieser Gruppe einmal etwas genauer, so findet man mit den Befehlen IN (C),r und OUT (C),r eine elegante Möglichkeit, mehr als die 256 ursprünglichen Ports zu adressieren. Bei diesen Befehlen wird der Zustand der unteren 8 Adressbits vom Inhalt des C-Registers bestimmt, zusätzlich wird aber der Inhalt von B auf die Adressbits A8 bis A15 gelegt. Damit stehen insgesamt 65536 Portadressen zur Verfügung.
Genau diese Eigenschaft des Z80 haben die Entwickler des CPC 464 genutzt. Alle Peripherie-ICs werden mittels der Adressbits A8 bis A15 selektiert.
Solche Tricks haben leider oft einen Haken. In diesem Fall besteht der Haken in einer deutlichen Beschränkung des Befehlsvorrates des Z80. Alle anderen I/O-Befehle des Z80 sind nicht mehr einsetzbar. Besonders gilt dies für die I/O-Befehle mit Schleifenautomatik. Sie benutzen das B-Regi-

ster als Zähler und stehen daher nicht als 'Lieferant' für das Highbyte der Portadresse zur Verfügung. Speziell sind das die Befehle INI, INIR, IND und INDR sowie OUTI, OTIR, OUTD und OTDR.

Als dritte Besonderheit des CPC 464 ist die Verwendung der Wait-Zyklen zu sehen.
Die Notwendigkeit dieses Anschlusses am Z80 resultiert noch aus der Zeit, als die verfügbaren Speicher-ICs recht gemütliche Gesellen waren. Besonders die ersten EPROMs ließen sich nach Anlegen der Adresse bis zu einer µs Zeit, bis sie die Daten parat hatten.
Um den Z80 mit solchen 'Langweilern' zu betreiben, war es nötig, eine bestimmte Zeit zu warten. Diese Wartezeit kann durch das Signal WAIT* erzeugt werden. Nach jeder negativen Flanke am Takt-Eingang überprüft der Prozessor den Zustand des Wait*-Anschlusses. Liegt dieser Anschluß auf 0 Volt, dann fügt der Z80 einen sogenannten Wait-Zyklus von der Dauer des Taktes ein. Nach Ablauf des Taktsignals, also mit der negativen Flanke, wird wieder der Zustand der Wait*-Leitung geprüft u.s.w.
Das im CPC dies Signal verwendet wird, liegt aber nicht an den verwendeten Speicher-ICs. Die sind allemal schnell genug für einen Z80 mit 4 MHz. Der Grund ist die nötige Synchronisation zwischen Prozessor und Video-Controller. Da beide ICs auf den Speicher zugreifen können, muß eine Kontrolle darüber bestehen, wer zu welcher Zeit an der Reihe ist. Dabei hat der Video-Controller unbedingten Vorrang, da sonst die Anzeige auf dem Monitor stark gestört werden könnte. Um diese Synchronisation zu erreichen, wird zu jedem vierten Taktsignal für den Prozessor ein Wait*-Signal erzeugt. Obwohl der Prozessor mit 4 MHz (Mega Hertz = Millionen Schwingungen pro Sekunde) angesteuert wird, ergibt sich durch die Waitzyklen eine effektive Arbeitsfrequenz von ca. 3.3 MHz.
Nun ist diese Verlangsamung der Rechengeschwindigkeit an sich noch nicht so schlimm. Ärgerlicher ist die Tatsache, daß beim CPC die in den Datenblättern zum Prozessor zu findenden Angaben für die Befehlsausführungszeit nicht stimmen. Dadurch lassen sich exakte Zeitschleifen, wie sie z.B. bei der Verwendung spezieller, besonders schneller Cassettenaufzeichnungsformate erforderlich sind, wohl nur schwer realisieren.

Die Signale BUSRQ* und BUSAK*, die Steuersignale für den DMA-Betrieb, sind im CPC nicht benutzt. Wohl aber sind sie auf den Expansion Connector geführt und stehen hier externen Erweiterungen zur Verfügung.
Auch das Signal HALT* wird im CPC nicht verwendet, ist aber ebenfalls am Expansion Connector verfügbar.

1.3 Das Gate Array, der System-Koordinator

Fast alle Bauteile im CPC sind handelsüblich. Man kann sie in jedem gut sortierten Elektronik-Laden erwerben. Die einzigen Ausnahmen sind das Rom und das Gate Array, im Schaltplan als IC116 bezeichnet. Das zuletzt genannte IC soll uns in diesem Abschnitt beschäftigen.
Dieses 40-polige IC ist speziell für den CPC entwickelt worden und hat mehrere wichtige Aufgaben. Wollte man alle integrierten Funktionen mit TTL-Gattern nachbilden, die Anzahl der ICs des CPC würde sich schnell mehr als verdoppeln.
Die Aufgaben des Arrays sind unter anderem:

Erzeugung aller benötigten Taktfrequenzen
Erzeugung der Signale für den Betrieb des dynamischen Rams
Steuerung der Zugriffe auf das Ram
Ab- und Zuschalten des Rom in den Speicherbereich
Erzeugung der Video-Signale
Erzeugung der RGB-Informationen für den Farb-Monitor
Steuerung des Bildschirmmodus
Speicherung der Tinten-Farben
Erzeugung des Interrupt-Impulses

Leider sind über dieses interessante IC nur sehr wenig Informationen verfügbar. Ein Datenblatt oder eine vergleichbare Beschreibung sind nirgends erhältlich, da der Hersteller das Innenleben wohl als Betriebsgeheimnis ansieht.
Unsere Bemühungen und Versuche, die Funktion des ICs möglichst gründlich zu erforschen, waren allerdings recht erfolgreich. So wollen wir Ihnen unsere Ergebnisse nicht vorenthalten.

1.3.1 Die Anschluß-Belegung des Gate Array

Das alles bestimmende Signal des CPC ist das am Pin 8 (XTAL) anliegende Quarzsignal mit einer Frequenz von 16 MHz. Das IC125, ein TTL-IC vom Typ 7400, bildet mit zwei seiner vier Gatter eine typische Oszillatorschaltung. Dieses Signal stellt quasi den Her(t)zschlag des CPC dar.

Die Eingangsfrequenz durch vier geteilt steht als Taktsignal von 4 MHz am Pin 39 als Takt Ø für den Prozessor zur Verfügung.

Ein weiteres Mal durch vier geteilt ergibt sich eine Frequenz von 1 MHz. Dieses Signal wird am Pin 1 des Gate Array zur Verfügung gestellt.
Das 1-MHz-Signal hat zwei Verwendungen. Zum einen ist es das Taktsignal für den Sound Chip, zum anderen bestimmt es mit, ob der Prozessor oder der CRTC das RAM adressieren kann. Bei einem Low werden die Adressleitungen des Prozessors über die Multiplexer IC104, 105, 109 und 113 zum Ram geschaltet.

CPU ADDR*	1	MA0/CCLK
READY		Ø
CAS*		Vcc1
244EN*		RESET*
MWE*		R
CAS ADDR*		GND
RAS*		G
XTAL		Vcc2
Vcc2		B
INTERRUPT*		D 7
SYNC*		D 6
ROMEN*		D 5
RAMRD*		D 4
HSYNC		D 3
VSYNC		D 2
IORQ*		D 1
M1*		D 0
MREQ*		DISPEN
RD*		Vcc1
A 15		A 14

1.3.1.1 Pinout des Gate Array

Da allerdings die Ansteuerung des Ram im CPC nicht ohne Tücken ist, finden Sie eine ausführliche Beschreibung der Ram-Steuersignale in einem späteren eigenen Kapitel.

Da die Ram-Bausteine nur über 8 Adressleitungen verfügen, muß die gesamte 16-Bit-Adresse gemultiplext, also zeitlich nacheinander an die Eingänge gelegt werden. Diese zeitliche Steuerung wird mit den Signalen CAS ADDR* (Pin 6), CAS* (Pin 3) und RAS* (Pin 7) erreicht. Die Signale RAS* und CAS* werden direkt an die Rams gelegt, das Signal CAS ADDR* wird an die schon erwähnten Multiplexer geführt.

Auch das Signal MA0/CCLK an Pin 40 des Gate Array hat eine Frequenz von 1 MHz. Dieses Signal ist allerdings zum CPU ADDR*-Signal phasenverschoben, d.h. die beiden Frequenzen sind zu unterschiedlichen Zeiten High. MA0/CCLK hat ebenfalls eine Doppelfunktion. Einmal stellt es das Taktsignal für den CRTC dar, der von diesem Signal alle weiteren Signale ableitet, zum zweiten wird es als Hilfsadressbit an den Adress-Multiplexer IC106 gelegt. Die Funktion dieses Hilfsadressbits wird ebenfalls später bei der Ansteuerung der Rams genauer erörtert werden.

Des weiteren wird vom Gate Array das Signal RAMRD* am Pin 13 erzeugt. Dieser Anschluß wird dann Low, wenn der Prozessor nach Anlegen einer Adresse Daten aus dem Ram lesen will und dies durch sein RD* -Signal dem Array an Pin 19 mitteilt. Da sich in weiten Bereichen Rom und Ram überlagern, kann das RD*-Signal des Prozessors nicht direkt verwendet werden. Sollen Daten aus dem Rom gelesen werden, so bleibt das Signal RAMRD* High und die Ausgänge des IC114, ein sogenannter Puffer/Zwischenspeicher, werden hochohmig. Dadurch kann zu diesen Zeiten keine Information vom Ram auf den Datenbus gelangen, obwohl die Speicheradresse auch an das Ram gelangt ist und es an seinen Ausgängen ein Byte bereitstellt.
Zusätzlich zum RAMRD* wird das READY-Signal vom Pin 2 des GA an das IC114 gelegt. Dieses Signal erzeugt am Prozessor das Signal zum Einfügen der Wait-Zyklen. Durch die zusätzliche Verschaltung des READY mit dem IC114 wird erreicht, daß sich die Information auf dem Prozessor-Datenbus während der Wait-Zyklen nicht ändert. Der 74LS373 speichert nach Anlegen eines High am Pin 11 die derzeitige Ausgangsinformation, bis dieser Anschluß wieder Low wird. Danach verhält sich das IC wie ein einfacher Puffer, d.h. die Ausgänge folgen den Änderungen der Eingänge unmittelbar.
Das Signal ROMEN* an Pin 12 des GA wird Low, wenn der Prozessor Daten aus dem Rom lesen will. Das im CPC eingebaute 32k-Rom belegt die Adress-Bereiche von &0000 bis &3FFF und von &C000 bis &FFFF. Es ist also in zwei unabhängigen Hälften ansprechbar. Ob in den sich überlagernden Speicherbereichen aus Rom oder Ram gelesen werden soll, muß dem GA über einen OUT-Befehl mitgeteilt werden. Dabei ist es durchaus möglich, nur eine Hälfte des Rom zu aktivieren.

Entsprechend der gewünschten Speicherkonfiguration dekodiert der GA den Zustand der Adressleitungen A14 und A15. Je nach gefordertem Speicher wird dann beim Lesen das RAMRD*- oder ROMEN*-Signal aktiv.

Ein Schreibbefehl des Prozessors geht unabhängig von der gewählten Speicherkonfiguration immer ins Ram. Dazu wird vom GA das Signal MWE* erzeugt.

Zusätzlich zur beschriebenen Funktion werden die Adressleitungen A14 und A15 an den Pins 20 und 21 aber noch für einen anderen Zweck verwendet. Auch das GA hat eine Portadresse, die benutzt wird, um die verschiedenen Möglichkeiten des GA zu programmieren. Die Portadresse ist &7F00 und wird über die Adressleitungen (A14 High, A15 Low) und das Signal IORQ* an Pin 18 decodiert.

Da der Datenbus des Z80 nicht direkt mit den Datenleitungen D0 bis D7 des GA verbunden ist, legt das Array den Anschluß 244EN* auf Low, wenn die Portadresse &7F00 in der zuvor beschriebenen Art erkannt wird. Dadurch werden die Ausgänge des IC115 (74LS244, ein Datenbuspuffer) freigegeben und das vom Z80 gelieferte Byte kann in das Array geschrieben werden.

Aber auch das Signal IORQ* hat für den GA eine Doppelbedeutung. Eine spezielle Eigenart des Z80 ist es, bei einem erkannten Interrupt gleichzeitig die Signale IORQ* und M1* auf Low zu legen. Dieser Zustand wird vom GA erkannt und der Interrupt-Impuls wird sofort gelöscht. Ist dagegen durch den Befehl DI, Disable Interrupt, die Behandlung des IRQ ausgeschaltet, so bleibt der Anschluß 10 des GA bis zum Wiederzulassen des IRQ Low. Sobald der IRQ mit EI wieder eingeschaltet ist, wird der anliegende Interrupt erkannt und der Interrupt-Ausgang wird wieder High.
Erzeugt wird das Interrupt-Signal an Pin 10 durch eine programmierbare Teilerkette im GA. Diese Teilerkette wird mit dem CRTC-Signal HSYNC versorgt und teilt die anliegende Frequenz durch 52. Da der HSYNC-Impuls ca. alle 65 Microsekunden auftritt, ergibt sich eine Zeit von 3,3 Millisekunden zwischen zwei Interruptimpulsen. Dabei werden die Impulse mit dem VSYNC-Signal des CRTC gekoppelt. Die Breite des VSYNC ist im CRTC auf ca. 500 Microsekunden programmiert. Nach etwa 125 Microsekunden erscheint der Interrupt, somit bleibt der Interrupt-Routine noch etwa 375 Microsekunden Zeit, am Portbit 0 des Port B des 8255 zu prüfen, ob ein VSYNC vorhanden ist. Dieses Signal wird als Zeitgeber bei verschiedenen Operationen benutzt.
Dieser Fall tritt aber nur bei jedem fünfzehnten Interrupt auf, bei den restlichen 14 Abfragen ergibt sich ein High des VSYNC, der interne Zähler wird nicht beeinflußt.

Die Signale HSYNC und VSYNC werden aber natürlich wie auch DISPEN zum Erzeugen des Video-Signals benötigt. Eine Verknüpfung dieser Signale ergibt das SYNC*-Signal am Pin 11 des GA.

1.3.2 Der Registeraufbau des Gate Array

Um alle beschriebenen Aufgaben ausführen zu können, müssen Daten im GA gespeichert werden. Die genaue Anzahl der internen Register ist nicht bekannt, allerdings können wir die vermutlich wichtigsten Register beschreiben.

Wie auch alle anderen Bausteine im CPC wird das GA über die Port-Adressierung angesprochen.
Die belegte Adresse ist &7Fxx. Daraus resultiert, daß das Adressbit A15 Low, das Adressbit A14 dagegen High sein muß. Die übrigen Adressbits (A12 bis A8) müssen gesetzt (auf High-Pegel) sein, da die anderen Peripherie-Bausteine in ähnlich unvollständiger Weise decodiert werden. Bei diesen Bausteinen sind die Selektionseingänge auch nur mit einzelnen Adressbits verbunden.
Der Zustand des unteren Adressbytes ist für die Dekodierung unerheblich, hier kann jeder beliebige Wert anliegen.

Insgesamt kann zwischen drei verschiedenen Registern unterschieden werden.
Die ersten beiden Register stehen im Zusammenhang mit der Farberzeugung, genauer mit den durch PEN und INK festgelegten Farbzuordnungen.
Das erste Register wird mit der Adresse geladen, in die ein Farbwert eingeschrieben werden soll. Wir wollen es weiterhin als Farbnummer-Register (FN-Reg) bezeichnen.
Den eigentlichen Farbwert kann man danach in das zweite Register (unter derselben Portadresse!) einschreiben. Dieses Register werden wir als Farbwert-Register (FW-Reg) bezeichnen.

Das dritte Register ist ein Multifunktionsregister (MF-Reg) und bestimmt den Bildschirmmodus und die Speicherkonfiguration. Dabei wird die Auswahl der verschiedenen Möglichkeiten durch die einzelnen Bits innerhalb des Registers bestimmt.

Alle Register des GA lassen sich nur beschreiben. Ein Auslesen der Werte ist NICHT möglich.

Da das GA nur über eine einzige Portadresse angesprochen werden kann, muß es einen Weg geben, zwischen den verschiedenen Gruppen zu unterscheiden. Diese Unterscheidung wird mit den beiden obersten

Bits des Datenbytes festgelegt. Die möglichen Kombinationen lauten:

Bit7	Bit6	
0	0	Wert in das **FN-Reg** schreiben
0	1	Farbwert in das gewählte **FW-Reg** schreiben
1	0	Wert in das **MF-Reg** schreiben
1	1	nicht genutzt ?

Was hat es aber nun mit den Farbnummer- und Farbwert-Registern auf sich?
Im Grunde stellen diese beiden Register die Entsprechungen zu den Basic-Befehlen PEN und INK dar. Der PEN-Befehl wird bekanntermaßen benutzt, um die aktuelle Schreibfarbe auf dem Monitor zu verändern. Die Zuordnung einer PEN-Nummer zu einer Farbe kann mit dem INK-Kommando festgelegt werden. Dazu wird die zu ändernde Nummer und der gewünschte Farbwert angegeben. Genau diese Funktionen werden mit den beiden Registern ausgeführt. In das FN-Register wird die Nummer der zu ändernden Farbe eingetragen, danach kann der gewünschte Farbwert in das GA geschrieben werden.
Um z.B. die zu PEN 1 gehörende Farbe zu ändern, ist der folgende Ablauf nötig:

OUT &7F00,&X00000001 : OUT &7F00,&X010XXXXX

Im ersten OUT-Befehl sind die Bits 6 und 7 = 0. In den Bits 0 bis 3 wird die Nummer der zu ändernden Farbe angegeben. In unserem Beispiel ist das die Nummer 1. Das Bit 5 hat keine Funktion, das Bit 4 hat eine besondere Bedeutung, auf die wir gleich noch zu sprechen kommen.
Im zweiten OUT-Befehl sind die Bits 6 und 7 so gewählt, daß das FW-Register angewählt ist. Die als 'X' bezeichneten Bits bestimmen nun den Farbenwert. Mit 5 Bit sind zwar 32 verschiedene Farben möglich, aber nur 27 verschiedene Farben werden erzeugt. Die verbleibenden 5 Farben sind mit anderen Farben identisch.

Wenn Sie dieses Beispiel in Basic ausprobieren, werden Sie feststellen, daß sich der gewünschte Erfolg nicht so recht einstellt. Ein kurzes Aufblitzen der neuen Farbe ist alles, was dabei herauskommt.
Ursache ist eine Eigenart der Software des CPC. Grundsätzlich werden alle Farben 'blinkend' dargestellt. Das bleibt aber unbemerkt, da nicht zwischen verschiedenen, sondern gleichen Farben umgeschaltet wird. Bei jedem Umschalten der Farben werden alle Parameter für das GA neu geladen. Wenn Sie aber vor die OUT-Kommandos den Befehl '**SPEED INK 255,255**' setzen, dann können Sie zumindest bei einigen Versuchen eine deutlich längere Zeit die Auswirkung betrachten.

Doch jetzt die Erklärung des bisher ausgesparten Bit 4 im FN-Reg. Ist dieses Bit beim Zugriff auf das Register gesetzt, dann wird die Information in

den Bits 0 bis 3 ignoriert, der im nächsten OUT-Befehl übermittelte Farbwert wird als neue Rahmenfarbe interpretiert.

Das MF-Register wird adressiert, wenn im OUT-Befehl das Bit 7 gesetzt und das Bit 6 Low ist. Die übrigen Bits dieses Registers haben folgende Bedeutung:

Bit 5 : Dies Bit hat keine Funktion ?
Bit 4 : 1 = V-Sync-Zähler löschen
Bit 3 : 1 = ROM &C000 bis &FFFF abschalten
Bit 2 : 1 = ROM &0000 bis &3FFF abschalten
Bit 1 : Bildschirm-Modus
Bit 0 : Bildschirm-Modus

Über die Funktion des Bit 5 in diesem Register konnte bisher nichts in Erfahrung gebracht werden.

Ist das Bit 4 gesetzt, so wird die Teilerkette für den Interruptimpuls gelöscht und der Zählvorgang der V-Sync-Impulse beginnt von neuem. Auf diese Weise könnte der zeitliche Abstand zwischen zwei Interruptimpulsen verlängert werden. In Basic können Sie sich von der Funktion mittels der folgenden kleinen Programmschleife überzeugen:

10 OUT &7F00 , &X10010110 : GOTO 10

Nach dem Start der Programmzeile ist der Rechner vollständig blockiert. Auch ein Reset über SHIFT/CTRL/ESC ist nicht mehr möglich. In diesem Einzeiler wird das Zähl-Register so schnell gelöscht, daß überhaupt keine Interrupt-Impulse mehr auftreten können. Da aber die Tastatur bei jedem Interrupt abgefragt wird, hilft nur noch das Aus- und wieder Einschalten, um den CPC wieder bedienbar zu machen.

Die Bits 2 und 3 bestimmen die momentane Speicherkonfiguration. Ist eins der Bits gesetzt, so befindet sich für den Prozessor in den angegebenen Adressbereichen bei Lesezugriffen das Ram, sind die Bits gelöscht, dann liest der Prozessor die Daten aus dem Rom.
Diese beiden Bits planlos zu manipulieren, führt mindestens zu Fehlermeldungen, Systemabstürze oder ein Reset sind aber genau so möglich.

Die verbleibenden Bits 0 und 1 bestimmen den aktuellen Bildschirm-Modus. Die möglichen Kombinationen sind:

Bit1	Bit0	
0	0	**Mode 0,** 20 Zeichen/Zeile, 16 Farben
0	1	**Mode 1,** 40 Zeichen/Zeile, 4 Farben
1	0	**Mode 2,** 80 Zeichen/Zeile, 2 Farben
1	1	wie Mode 0, aber kein Blinken

Wenn Sie den Einzeiler zum Ausschalten des Interrupt im Mode 1 probiert haben, so werden Sie eine seltsame Veränderung der Zeichen auf dem Bildschirm festgestellt haben. In diesem Beispiel haben wir als Bildschirm-Modus den 80-Zeichen-Modus gewählt und ohne den Bildschirm zu löschen umgeschaltet. Die dargestellten Zeichen sehen aus, als ob in der Mitte jedes Zeichens Punkte fehlen. Die Erklärung zu diesem Phänomen finden Sie im Anschluß an das folgende Kapitel, wenn der Aufbau des Bildschirms und die Darstellung der Zeichen beschrieben wird.

1.4 Der Video–Controller HD 6845

Die Hauptarbeit bei der Erzeugung des Bildes auf dem Monitor leistet der Video–Controller HD 6845, der auch als Cathode Ray Tube Controller, kurz CRTC bezeichnet wird. Dieses IC wurde speziell als Interface zwischen Microprozessoren und Rasterbildschirmen wie den üblichen Monitoren entworfen.
Er erzeugt aus einem einzigen Taktsignal alle für den Monitor erforderlichen Synchronsignale, wobei sich alle benötigten Parameter in weiten Grenzen programmieren lassen.

Bevor wir die Anschlußbelegung und den internen Registeraufbau beschreiben, wollen wir einen kurzen Überblick über die Möglichkeiten dieses interessanten Bausteins geben.

Programmierbare Anzahl der Zeichen pro Zeile
Programmierbare Anzahl der Zeilen pro Bildschirm
Programmierbare vertikale Punktmatrix der Zeichen
Zugriff auf einen Speicherbereich von 16 K
Automatischer Refresh bei Verwendung dynamischer Rams
Cursor–Controll–Funktionen
Programmierbarer Cursor (Höhe und Blinken)
Light–Pen–Eingang
Einfache 5 Volt–Betriebsspannung
TTL–kompatible Ein– und Ausgänge

Urspünglich wurde der 6845 von Motorola für den Einsatz in Computersystemen mit der Prozessor–Familie 68xx entwickelt. Auf Grund der außergewöhnlichen Flexibilität und einfachen Handhabung ist dieser Controller aber in sehr vielen Systemen finden. Selbst bei so leistungsstarken Systemen wie z.B. Sirius ist dies IC zu finden.

1.4.1 Die Anschlüsse des CRTC

Die Bedeutung der 40 Anschlußbeine ist wie folgt:

MA0 – 13 : **Memory Adress Lines**, über diese 14 Anschlüsse werden die Speicherplätze des Bildspeichers adressiert.

RA0 – 4 : **Raster Adress Lines**, diese 5 Anschlüsse wählen aus dem Charactergenerator die derzeitige Rasterzeile des darzustellenden Zeichens aus.

D0 – 7 : **Bidirectional Data Bus**, über diese Pins werden Informationen in den Controller geschrieben und aus ihm herausgelesen.

Vss (1)	VSYNC
RES*	HSYNC
LPSTB	RA 0
MA 0	RA 1
MA 1	RA 2
MA 2	RA 3
MA 3	RA 4
MA 4	D 0
MA 5	D 1
MA 6	D 2
MA 7	D 3
MA 8	D 4
MA 9	D 5
MA 10	D 6
MA 11	D 7
MA 12	CS*
MA 13	RS
DISPTMG	E
CUDISP	R/W*
Vcc	CCLK

1.4.1.1 Pinout des CRTC HD 6845

R/W* : **Read/Write***, dieses Signal bestimmt die Datenrichtung auf den Datenleitungen. Bei einem Low können Daten vom Prozessor in den CRTC geschrieben werden, bei High werden sie aus dem CRTC gelesen.

CS* : **Chip Select***. Um Datentransfer mit dem 6845 zu ermöglichen, muß er adressiert werden. Dies geschieht durch ein Low am CS*–Eingang.

RS : **Register Select**. Dieses Signal wird benötigt, um zwischen Adress–Register und 18 Controll–Registern zu selektieren. Bei Lowpegel an RS kann auf das Adress–Register zugegriffen werden, bei einem High besteht Zugriff auf die Controll–Register.

EN : **Enable**. Mit der steigenden Flanke dieses Signals werden die am IC anliegenden Prozessorsignale vom Controller übernommen.

RES* : **Reset***, ein Low an diesem Eingang setzt alle Zähler im CRTC zurück und alle Ausgänge werden Low.
Diese Funktion wird aber nur ausgeführt, wenn gleichzeitig der LPSTB–Eingang Low ist. Der Reset löscht nicht die Controll–Register!

CLK : **Character Clock** ist das Taktsignal, aus dem alle vom Monitor benötigten Signale durch Teilung abgeleitet werden.

HSYNC : **Horizontal Sync** liefert das Signal für die horizontale Synchronisation des Monitors. Falsch eingestellter oder fehlender HSYNC äußert sich im 'Durchlaufen' des Bildes.

VSYNC : **Vertical Sync** liefert das vom Monitor benötigte Signal zur vertikalen Synchronisation.

DISPTMG : **Display Timing**. Dieses Signal ist zu den Zeiten High, wenn das dem Monitor zugeführte Signal auf dem Bildschirm darzustellen ist. Mit Hilfe dieses Signals lassen sich die Strahlrückläufe unterdrücken.

CUREN : **Cursor Enable** (oft auch als Cursor Display, CURDISP, bezeichnet) wird verwendet, wenn der Cursor nicht durch die Software, sondern den CRTC selbst gesteuert wird. Auch das Cursorblinken kann mit diesem Anschluß gesteuert werden.

LPSTB : **Light Pen Strobe**. Wird an diesem Eingang eine Low—High—Flanke angelegt, dann wird der derzeitige Zustand der MA—Leitungen in die Light—Pen—Register übertragen und gespeichert. Diese Register können ausgelesen und in einem entsprechenden Programm verwendet werden.

1.4.2 Die internen Register des Video—Controllers

Wie bereits erwähnt, enthält der 6845 ein Adress—Register und 18 Controll—Register. Da mit dem Signal RS, Register Select, aber nur zwischen zwei Adressen ausgewählt werden kann, stellt sich die Frage, wie alle 18 Controll—Register über nur eine Adresse angesprochen werden können. Die Lösung des Problems ist das Adress—Register. In das Adress—Register wird die Nummer des Controll—Registers geschrieben, auf das man als nächstes zugreifen möchte. Dieses Verfahren mutet zwar etwas umständlich an, hat aber einen unbestreitbaren Vorteil. Auf diese Art belegt der CRTC eben nur zwei und nicht 18 oder gar 32 Adressen. Da außerdem der CRTC normalerweise nur einmal beim Einschalten des Gerätes programmiert wird, kann auch der Mehraufwand an Programmierung in Kauf genommen werden.

Aber betrachten wir nun die 18 Register etwas detaillierter. Die folgende Beschreibung fällt allerdings wegen der komplexen Struktur einzelner Register etwas trocken und schwer verständlich aus. Auch sind zum Verständnis einiger Register grundlegende Kenntnisse der Videotechnik nötig. Sollten Sie beim Lesen nicht alles verstehen, so trösten Sie sich mit der Gewißheit, daß der Videocontroller in Ihrem Computer nicht 'von Hand' programmiert werden muß.

In der folgenden Aufstellung bedeutet ein R hinter der Registerbezeichnung, daß dieses Register zu lesen ist, ein W bedeutet die Möglichkeit, dieses Register zu beschreiben. Beachten Sie, daß einige Register nur zu beschreiben oder zu lesen sind (gekennzeichnet durch —).

AR —/W : **Adress Register**. Dieses 5—Bit—Register wird mit der Nummer des gewünschten Controll—Registers geladen. Registerwerte 18 bis 31 werden ignoriert, die gültigen Werte lauten 0 bis 17. Dieses Register wird angesprochen, wenn sowohl CS als auch RS Low sind.

R0 —/W : **Horizontal Total**. In dieses 8—Bit—Register wird die Anzahl der Zeichen pro totaler Zeile eingetragen. Allerdings ist eine totale Zeile wesentlich länger als die am Bild-

schirm sichtbaren Zeichen, da auch die Zeiten für den Rand und den Strahlrücklauf mitgerechnet werden müssen. Entsprechend wird dieser Wert etwa 1.5 mal so groß wie die Anzahl der dargestellten Zeichen gewählt sein.

R1 -/W : **Horizontal Displayed**. Dieses Register enthält die Anzahl der am Bildschirm darzustellenden Zeichen. Der hier eingetragene Wert muß kleiner als der von R0 sein.

R2 -/W : **Horizontal Sync Position**. Der 8–Bit–Wert dieses Registers bestimmt den Zeitpunkt des HSync–Impulses. Wird der Wert von R2 verringert, so verschiebt sich das Monitorbild nach rechts, eine Erhöhung schiebt das Bild nach links.

R3 -/W : **Sync Width**. Mit den unteren 4 Bit dieses Registers wird die Breite der HSync und VSync–Impulse festgelegt. Die oberen 4 Bit dieses Registers werden nicht benutzt.

R4 -/W : **Vertical Total**. Die unteren 7 Bit dieses Registers bestimmen die Anzahl aller Rasterzeilen pro Bild. Der Wert bestimmt damit auch, ob eine Bildwiederholfrequenz von 50 oder 60 Hertz gewählt wird.

R5 -/W : **Vertical Total Adjust**. Mit Hilfe der unteren 6 Bit dieses Registers kann ein Feinabgleich der Bildwiederholfrequenz vorgenommen werden.

R6 -/W : **Vertical Displayed**. Die unteren 7 Bit dieses Registers bestimmen die Anzahl der tatsächlich dargestellten Rasterzeilen auf dem Monitor. Hier kann theoretisch jeder Wert programmiert werden, der kleiner als (R4) ist.

R7 -/W : **Vertical Sync Position**. Der 7–Bit–Wert dieses Registers bestimmt den Zeitpunkt des VSync–Impulses. Wird der Wert von R7 verringert, so verschiebt sich das Monitorbild nach unten, eine Erhöhung schiebt das Bild nach oben.

R8 -/W : **Interlace**. Mit den unteren beiden Bits dieses Registers wird bestimmt, ob die Darstellung mit oder ohne Zeilensprung–Verfahren (Interlace) erfolgen soll.

R9 -/W : **Maximum Raster Adress**. Dieses 5–Bit–Register bestimmt die Anzahl der Rasterzeilen der darzustellenden Zeichen.

R10 -/W : **Cursor Start Raster**. Die Bits 0 bis 4 dieses Registers bestimmen, auf welcher Rasterzeile der Cursor beginnen soll. Die Bits 5 und 6 legen den Cursormodus fest. Der Cursormodus wird dabei mit den Bits wie folgt festgelegt:

Bits 6	5	
0	0	Cursor nicht blinkend
0	1	Cursor nicht dargestellt
1	0	Cursor blinkt (ca 3 x pro Sek.)
1	1	Cursor blinkt (ca 1.5 x pro Sek.)

R11 -/W : **Cursor End Raster**. Entsprechend zu (R10) legen die unteren 5 Bit dieses Registers fest, auf welcher Rasterzeile der Cursor endet.

R12 R/W : **Start Adress High**. Die Bits 0 bis 5 legen fest, ab welcher Adresse im gesamten 16k–Adressbereich des CRTC der Bildspeicher beginnt. Wird dieses Register gelesen, so sind die Bits 6 und 7 immer Low.

R13 R/W : **Start Adress Low**. Dieses Register legt analog zu (R12) das niederwertige Adressbyte des zu adressierenden Bildschirmspeichers fest.

R14 R/W : **Cursor High**. Die Bits 0 bis 5 dieses Registers stellen das High–Byte der momentanen Cursorposition dar.

R15 R/W : **Cursor Low**. Analog zu (R14) wird in diesem Register das Low–Byte der Cursor–Adresse abgelegt.
Da sowohl R14 als auch R 15 beschrieben und gelesen werden können, kann über diese Register die Cursorposition frei bestimmt werden.

R16 R/- : Dieses Register enthält nach einem positiven Strobeimpuls das Highbyte der zum Zeitpunkt des Impulses aktiven Bildschirmspeicheradresse. Die Bits 6 und 7 dieses Reisters sind immer Low.

R17 R/- : Analog zu R16 enthält dieses Register das Lowbyte zum Zeitpunkt des Light–Pen–Strobes.
Sowohl R16 wie auch R17 können nur gelesen werden.

1.5 Das Ram des CPC

Die im CPC eingebauten 64 K Ram (Schreib/Lesespeicher) werden nicht nur als Daten- und Programmspeicher eingesetzt. Auch die Bildschirminformation wird in diesem Speicher untergebracht.

Nachdem in den vorherigen Kapiteln die drei wichtigsten Bausteine des CPC 464, der Prozessor, das Gate Array und der Video-Controller, detailliert besprochen wurden, werfen wir in diesem Abschnitt einen Blick auf das Zusammenspiel dieser drei Komponenten beim Zugriff auf die Speicher-ICs. Dabei wird auch geklärt, wie der Video-Controller das Ram anspricht, um Zeichen auf dem Bildschirm darzustellen.

Zuvor aber wollen wir einen kleinen Abstecher machen und uns anschauen, wie die verwendeten dynamischen Ram-Bausteine überhaupt funktionieren.

Als erstes soll einmal geklärt werden, wie die Adressierung von 65536 Speicherzellen mit den zur Verfügung stehenden 8 Adress-Anschlüssen möglich ist.
Das grundsätzliche Funktionsprinzip besteht darin, die 16-Bit-Adresse in zwei Hälften zu teilen und diese beiden Adress-Bytes nacheinander an die Adress-Pins der Rams zu legen. Dieser Vorgang wird Multiplexen genannt. Allerdings erfordert das Multiplexen Steuersignale, die den Rams mitteilen, welche Information zur Zeit an den Adressanschlüssen anliegt.

An diesem Punkt kommen die vom Gate Array gelieferten Signale RAS* und CAS* ins Spiel.
Nachdem ein Adress-Byte an den Rams anliegt, wird ihnen durch einen High-Low-Wechsel des Signals RAS* mitgeteilt, daß eine Adresshälfte parat ist. Mit der negativen Flanke (dem High-Low Wechsel) des RAS* wird die anliegende Adressinformation in den Rams gespeichert.
Jetzt kann die zweite Hälfte der Adresse an das Ram angelegt werden. Sobald dieses Adressbyte anliegt, wird das CAS*-Signal Low. Damit hat das Ram die gesamte 16-Bit-Adresse erhalten und wählt die gewünschte Speicherzelle an. Diese Zelle kann jetzt beschrieben oder ausgelesen werden.
Die Umschaltung der Adresshälften muß natürlich auch von einem passenden Signal übernommen werden, im CPC ist es das Signal CAS-ADDR*.
Als Umschalter oder Multiplexer (meint beides dasselbe, Multiplexer hört sich nur viel fachmännischer an) arbeiten die ICs IC104, 105, 109 und 113. Die Funktion dieser ICs vom Typ 74LS153 kann man sich am besten wie zwei elektronisch gesteuerte Drehschalter vorstellen. Jeder der beiden Schalter hat vier Eingangsanschlüsse und einen Ausgang. Über zwei Steuereingänge kann entschieden werden, welcher der vier Eingänge mit dem Ausgang verbunden ist.

Die beiden Steuereingänge werden von den Signalen CPU-ADDR* und

CAS-ADDR* angesteuert. Mit dem Signal CPU-ADDR* wird entschieden, ob der Prozessor oder der CRTC eine Adresse an das Ram legen kann, CAS-ADDR* schaltet zwischen den jeweiligen Adresshälften um.

Die Umschaltung wollen wir uns exemplarisch am Multiplexer IC105 anschauen.
Die Ausgänge Pin 7 und Pin 9 sind über je einen Widerstand von 120 Ohm mit den Adresseingängen A0 und A1 der Rams verbunden.
Die Steuereingänge A (Pin 14) und B (Pin 2) sind mit den bekannten Signalen CPU-ADDR* und CAS-ADDR* verbunden.

Die Adressinformation liegt an den Pins 3 bis 6 und 10 bis 13. Hier finden Sie auch das im vorigen Kapitel mit Adresshilfsbit bezeichnete Signal CCLK wieder. Welches Adressbit bei welcher Steuerkombination an den Ausgängen erscheint, zeigt die folgende Aufstellung:

CPU-ADDR	CAS-ADDR	MULTIPLEXER-AUSGANG A0	MULTIPLEXER-AUSGANG A1
0	0	Z80, A9	Z80, A0
0	1	Z80, A2	Z80, A1
1	0	6845, MA8	CCLK
1	1	6845, MA1	6845, MA0

Nun trägt diese Tabelle auf den ersten Blick nicht besonders zum Verständnis der Ansteuerung des Rams bei. Besonders verwirrend ist es, daß die Adressleitung A0 des Prozessors nicht auf A0 des Rams liegt. Bedenken Sie aber, daß es dem Prozessor egal ist, in welche physikalische Adresse des Rams er seine Information schreibt. Für den Prozessor ist es z.B. beim Schreiben oder Lesen 'seiner' Speicherzelle 0 ohne Bedeutung, ob dabei auch wirklich die physikalische Ram-Adresse 0 oder eine beliebige andere Adresse im Ram adressiert wird. Er wird bei Zugriffen auf 'seine' Speicherzelle 0 immer wieder dieselbe Speicherzelle adressieren. Insofern ist die Bezeichnung der Adresspins der Rams eigentlich willkürlich und für den Prozessor unerheblich.

Viel wichtiger ist die Zuordnung von Prozessoradressen zu den Adressen des CRTC. Diese Zuordnung zeigt die Tabelle 1.5.0.1.

Wie man sieht, liegen alle Adressbits des Prozessors über die Multiplexer an den Adress-Anschlüssen der Rams, aber auch der Video-Controller adressiert unter Zuhilfenahme des CCLK den gesamten 64K-Speicherbereich. Das aber steht im Gegensatz zum vorigen Kapitel, wo ja gesagt wurde, daß der CRTC einen Bereich von nur 16K adressieren kann.
Diese Aussage ist insoweit richtig, da als Adressleitungen nur die 14 mit MA (Memory Adress Line) bezeichneten Anschlüsse gezählt werden. Diese 14 Anschlüsse ermöglichen einen Adressbereich von 16 K.

Z80	6845	Z80	6845
A0	CCLK	A8	MA7
A1	MA0	A9	MA8
A2	MA1	A10	MA9
A3	MA2	A11	RA0
A4	MA3	A12	RA1
A5	MA4	A13	RA2
A6	MA5	A14	MA12
A7	MA6	A15	MA13

1.5.0.1 Zugriff von Z80 und 6845 auf den gem. Speicher

Die im CPC eingesetzte Betriebsart des 6845 zur Adressierung des Video-Speichers wird nur selten verwendet. Mit den Anschlüssen RA0 bis RA4 wird normalerweise ein fest programmiertes Zeichen- oder Character-Rom angesteuert, das die Bitmuster für die auf dem Bildschirm darzustellenden Zeichen enthält.
Üblicherweise haben Computer einen als Video-Ram bezeichneten Speicherbereich, in dem die auf dem Bildschirm darzustellenden Zeichen gespeichert werden. In diesem Speicher belegt jede Zeichenposition ein Byte. Das ergibt z.B. bei der Darstellung von 80 x 25 Zeichen einen Speicherbedarf von 2000 Bytes.
Nun kann aber in einem Byte nicht die gesamte zur Darstellung benötigte Information untergebracht werden. Jedes Zeichen besteht ja aus einer Anzahl von untereinander liegenden Punktereihen.
Auch beim CPC 464 kann man diese Reihen auf dem Monitor erkennen. So besteht z.B. der Cursor aus 8 untereinander liegenden Reihen, in denen alle Bildpunkte 'an' sind. Bei der Darstellung von Buchstaben oder Ziffern sind nur bestimmte für die Darstellung des Zeichens erforderliche Punkte in einer Reihe an. Diese Punkte-Muster lassen sich durch Bitmuster speichern, wobei üblicherweise ein gesetztes Bit einem Punkt auf dem Bildschirm entspricht.
Die RA-Anschlüsse werden nun benötigt, um die einzelnen Reihen, also Bitmuster, aus dem Zeichen-Rom zu erhalten. Dazu werden die RA-Anschlüsse als Adressleitungen für das Zeichen-Rom verwendet.

Wie man sich vorstellen kann, ist es bei Verwendung von fest programmierten Zeichen-Roms nicht möglich, auf dem Bildschirm hochauflösende Grafik zu erzeugen. Nach diesem Prinzip konstruierte Computer sind an den eingebauten Zeichensatz gebunden.
Beim CPC entfällt aber dieses herkömmliche Character-Rom, hier hat man einen gänzlich anderen Weg beschritten.
Da die RA-Anschlüsse direkt den Speicher adressieren, muß also die Punkte-Information auch im Ram untergebracht sein. Nur durch diesen Schaltungstrick ist es möglich, jedes beliebige Bitmuster auf dem Monitor zu erzeugen, sprich Grafik in den bekannten Grenzen darzustellen.

Doch bevor wir uns dem konkreten Aufbau des Video-Speichers zuwenden, soll endlich das Signal CCLK erklärt werden. Dazu ist allerdings ein klein wenig Mathematik nötig.
Der CRTC wird mit einer Taktfrequenz von 1 MHZ angesteuert. Mit jedem Taktimpuls wird eine Speicherzelle adressiert. In dieser Zelle steht bitweise verschlüsselt die Information, welche Punkte auf dem Bildschirm 'an', also in der Schreibfarbe dargestellt sein sollen. Da eine Frequenz von 1 MHz einer Periodendauer von 1 μs entspricht, steht für die Darstellung jedes Punktes genau ein Achtel der Taktfrequenz zur Verfügung. Das ist eine Zeit von 0.125 μs. Um alle 640 Punkte einer Zeile darzustellen, ist somit eine Zeit von 80 μs erforderlich.
Da aber das die Dauer einer Zeile bestimmende V-Sync-Signal eine Periodendauer von 52 μs hat, kann diese Rechnung nicht aufgehen. Mit diesen Werten lassen sich maximal 40 Zeichen darstellen.
Ein Ausweg aus diesem Problem ist eine spezielle Betriebsart der Rams, der Page Adress-Mode. Hat ein Ram nach dem Anlegen der RAS- und CAS-Signale den Inhalt der gewünschten Speicherzelle auf die Datenausgänge gelegt, dann reicht es, mit einem weiteren CAS-Impuls nur eine neue Adress-Hälfte an die Rams zu legen, um das nächste Byte zu erhalten. Das setzt natürlich voraus, daß sich nur eine Hälfte der Adressinformation ändert.
Genau diese Eigenschaft haben die Entwickler des CPC genutzt. Natürlich muß die Adressinformation zu den beiden CAS-Impulsen unterschiedlich sein, sonst liest man dieselbe Speicherzelle zweimal. Das ist aber beim CCLK-Signal gegeben, es schaltet genau zwischen den beiden CAS-Impulsen um. Dieses Signal wird vom Multiplexer IC105 auf das Adressbit 0 (vom Prozessor aus gesehen) gelegt, wenn das Signal CAS-ADDR auf Low, das Signal CPU-ADDR dagegen auf High ist. Es stellt damit das unterste Adressbit des Video-Rams dar.
Die beiden schnell aufeinander gelieferten Bytes aus dem Video-Ram werden im Gate Array zwischengespeichert, in die für den Monitor benötigte serielle Form umgewandelt und zusammen mit den Farbinformationen an den RGB-Ausgang geliefert.

Bleiben noch die beiden Signale MA12 und MA 13. Mit Hilfe dieser beiden Bits wird innerhalb von 16k-Schritten der Beginn des Video-Rams bestimmt. Üblicherweise sind diese Bits gesetzt, das Video-Ram beginnt also bei &C000. Aber auch ein Video-Bereich von &4000 bis &7FFF ist bei entsprechender Programmierung möglich.

1.6 Das Video-Ram zwischen Z80 und 6845

Probieren Sie am CPC doch einmal dieses kurze Programm:

```
10  MODE 2
20  FOR i = &c000 TO &ffff
30  POKE i,255
40  NEXT i
```

Sie erhalten auf dem Bildschirm eine dünne Linie, die von der linken oberen Ecke schnell nach rechts gezeichnet wird. Am Ende der ersten Linie wird sie genau 8 Reihen tiefer fortgesetzt.
Ist der Bildschirm mit diesen dünnen Linien einmal gefüllt, so beginnt das Ganze wieder links oben, diesmal aber eine Punktereihe tiefer.

Probieren Sie das Programm auch einmal im MODE 1 und MODE 0.

Danach ändern Sie einmal die Zeile 30 in:

```
30  POKE i,1
```

Jetzt erhalten wir eine Punktereihe, die den Bildschirm zu senkrechten Reihen füllt.

Wenn das Programm im Mode 2 lief, dann sieht man, daß die senkrechten Reihen an der rechten Seite der Zeichen stehen.
Im Mode 1 erhalten wir zwei senkrechte Reihen pro Charakter, im Mode 0 sind es sogar 4.

Wir wollen eine letzte Änderung am Programm vornehmen. Löschen Sie dazu die Zeile 10 des Programms und geben Sie 'MODE 2' im Direktmodus ein. Der Bildschirm wird gelöscht und 'READY' erscheint in der linken oberen Ecke. Betätigen Sie die Cursor-Down Taste (Pfeil nach unten) bis die Ready-Meldung aus dem Bild verschwindet. Der Cursor steht jetzt auf der letzten Eingabezeile. Lassen Sie das Programm noch einmal laufen.

Das Ergebnis ist einigermaßen irritierend.

Dieses kleine Programm hat uns gleich mehrere wichtige Dinge verraten. Zum einen haben wir damit bewiesen, daß der Bildschirmspeicher bei &C000 beginnt und bei &FFFF aufhört. Überraschenderweise ist Lage und Größe der Bildschirmspeicher in allen drei Modi gleich. Es wird also nicht zwischen Modus 0 und Modus 2 unterschieden. Nur die erzeugten Farben sind unterschiedlich.

Allerdings gibt ein 16k-Bytes großer Bildschirmspeicher im Mode 0, also

bei 20 Zeichen pro Zeile offensichtlich wenig Sinn. 20 Zeichen mal 25 Zeilen ergibt nur 500 Zeichen auf dem Bildschirm. Warum benötigt der CPC scheinbar 16384 Speicherplätze, um diese 500 Zeichen darzustellen?

Die Antwort ist recht einfach. Wie bereits erwähnt besitzt der CPC keinen Video-Ram, in dem ein Zeichen in einem Byte gespeichert wird.
Im 80-Zeichenmodus belegt ein Zeichen auf dem Bildschirm 8 Bytes, bei 40 Zeichen sind es 16 Bytes und im 20-Zeichen-Modus ganze 32 Bytes. Das läßt sich auch aus dem Programm ersehen, welches die senkrechten Linien erzeugte.
Unsere Darstellung 1.6.0.1 macht den Aufbau eines Zeichens noch einmal deutlich. Dabei soll das Zeichen im Mode 2 in der linken oberen Bildschirmecke stehen.

Der 80-Zeichen-Modus ist in dieser Hinsicht am einfachsten zu verstehen, da ein gesetztes Bit einen Punkt in der aktuellen Zeichen- (Pen-) Farbe erzeugt. Ist ein Bit dagegen nicht gesetzt, so erscheint an dieser Stelle auf dem Bildschirm die Hintergrundfarbe. Da im Mode 2 nur eine Zeichenfarbe möglich ist, gibt es keine weiteren Möglichkeiten.

Wofür werden aber im Mode 0 32 Bytes für ein Zeichen benötigt?

Diese Zusammenhänge sind bei den Modi 0 und 1 nicht mehr so einfach zu beschreiben. Sie sollten das folgende kleine Programm einmal eintippen und die angezeigten Ergebnisse bei der Lektüre vor Augen haben. Dadurch werden die Beschreibungen sicher verständlicher, als wenn Sie einen reinen 'Trockenkurs' versuchen.

```
 10 MODE 2
 20 REM
 30 PRINT "A"
 40 FOR adress = &C000 TO &F800 STEP &800
 50  p$ = BIN$(PEEK(adress),8)
 60  FOR I = 1 TO 8
 70   IF MID$(p$,I,1) = "1" THEN PRINT "X"; ELSE PRINT ".";
 80  NEXT I
 90  PRINT
100 NEXT adress
```

Lassen Sie dieses Programm so wie beschrieben laufen, dann erhalten Sie ein Bild, das der abgedruckten Matrix des 'A' gleicht.

Ändern Sie nun einmal den Mode-Befehl in der Zeile 10 in 'MODE 1' und lassen Sie das Programm laufen. Das Ergebnis ist einigermaßen verblüffend.
Daß sich nur die halbe Matrix in den ausgelesenen Bytes befindet, war anzunehmen. Daß aber diese Matrix auch nur ein halbes Byte, also die

Bits 4 bis 7, beansprucht, verwirrt zunächst.

Wir kommen der Klärung des Rätsels aber näher, wenn Sie die Zeile 20 ersetzen:

20 PEN 2

Außer der geänderten Schreib-(PEN-)farbe hat sich auch das durch unser Programm angezeigte Bitmuster geändert. Das aber ist die Lösung unseres Problems!
Wenn Sie mit dem CPC bereits etwas vertraut sind, werden Sie wissen, daß im 40-Zeichen-Modus 4 Farben möglich sind. Diese vier Farben lassen sich einfach mit dem Zeichen selbst abspeichern, in dem nur vier Bit für die gesetzten Pixel maßgeblich sind, und Low- und High-Nibble (ein Nibble = ein Halb-Byte, 4 Bit) über die Farben entscheiden. Bei dem verwendeten Prinzip muß nur das Gate Array die Pixel für die Anzeige in horizontaler Richtung verdoppeln, um auch tatsächlich 8 Punkte darzustellen, wo nur vier Punkte gespeichert sind.

Im Mode 0 bei der Darstellung von 20 Zeichen pro Zeile wird diese Methode noch erweitert. Hier sind es nur zwei Bit, welche die Pixel-Information enthalten. Die Stellung der zwei Pixel innerhalb des Bytes bestimmen die Farbe, in der dieses Pixel dargestellt werden soll. Damit sind insgesamt 16 Kombinationen möglich, genau die Anzahl der zur Verfügung stehenden Farben. Da nur zwei Pixel in einem Byte gespeichert sind, werden 4 Byte für eine Pixel-Zeile benötigt, insgesamt also 8 x 4 = 32 Bytes für ein Zeichen in 16 verschiedenen möglichen Farben.

Probieren Sie doch einfach das Programm im Modus 0 mit verschiedenen Werten für das PEN-Kommando aus. Sie werden dann schnell hinter das Funktionsprinzip kommen.

Damit sind die beiden ersten Punkte vom Beginn des Kapitels geklärt. Unklar dagegen ist noch der Punkt der 'Verschiebung' des Bildschirmrams. Dieses Problem ist in der Hardware des CPC begründet.

Auch ein Z80 mit einer Taktfrequenz von 4 MHz benötigt zum Verschieben eines 16K-Datenblocks einige Zeit. Um z.B. beim Listen eines längeren Basicprogramms nicht für jede neue Zeile den gesamten Video-Ram-Bereich um 640 Speicher-Plätze zu verschieben, hat man eine spezielle Eigenschaft des CRTC genutzt. Durch entsprechende Programmierung der Register 12 und 13 des 6845 kann der Bildschirm praktisch auf jeder geraden Speicherzelle des Video-Rams beginnen. Dadurch kann das Scrollen sehr viel schneller passieren, da nur die entsprechenden Register mit den nötigen Werten versorgt werden müssen. Die neue Zeile am unteren Bildrand ist schnell gelöscht und mit den Zeichen versehen.

Ein Start des Video-Ram auf einer ungeraden Adresse, also z.B. bei &C001 ist wegen der beschriebenen Verwendung des Signals CCLK als Adressbit nicht möglich.

Das folgende Programm zeigt, daß eine Manipulation der genannten Register auch von Basic aus zu bewerkstelligen ist:

```
10 adrreg = &bc00  :  REM Adressregister des 6845
20 datreg = &bd00  :  REM Port des Datenregisters
30 OUT adrreg,13  :  REM Register wählen
40 FOR offset = 1 TO 40
50 OUT datreg,offset  :  REM 40 mal ändern
60 FOR warten = 1 TO 40  :  REM und etwas warten
70 NEXT warten,offset
```

In diesem Programm wird der Bildschirminhalt horizontal gescrollt. Ohne die Warteschleife würde das Scrollen so schnell ablaufen, daß man den Vorgang mit dem Auge gar nicht verfolgen könnte.

Auch vertikales Scrollen läßt sich von Basic aus programmieren. Allerdings müssen dann beide Register, Low- und Highbyte, manipuliert werden. Da aber zwischen den beiden OUT-Befehlen recht viel Zeit vergeht, kommt es zu unangenehmen Flimmer-Erscheinungen.

Es gibt beim Video-Ram aber noch eine Besonderheit zu beachten.

Rechnen wir die bekannten Werte einmal zusammen.
Im Mode 2 besteht ein Zeichen aus 8 Bytes. In einer Zeile haben 80 Zeichen Platz und es sind 25 Zeilen auf dem Bildschirm möglich. Das ergibt einen gesamten Speicherplatzbedarf von 80 x 25 x 8 = 16000 Bytes. Ein 16K-Speicherbereich hat aber 2 hoch 14 = 16384 Speicherplätze. Wo sind die fehlenden 384 Bytes?
Ganz einfach. Sie werden nicht benötigt. Jedenfalls nicht, so lange der Bildschirm nicht gescrollt wird.
Hier könnten kurzfristig zu speichernde Werte untergebracht werden, die aber spätestens beim nächsten CLS mit Sicherheit verschwunden sind.

Sie werden sich jetzt sicher fragen, wie um alles in der Welt mit dieser verrückten Organisation des Bildschirmspeichers jemals vernünftig Grafik programmiert werden kann.
Auch scheint es fast unmöglich, ein Zeichen vom Bildschirm zu lesen. Bei anderen Rechnern ist das kein Problem, da kann mit einem POKE ein Zeichen auf dem Bildschirm plaziert werden. Entsprechend kann der Inhalt des Video-Ram mit PEEK ausgelesen werden.
Weiterhin ist üblicherweise sicher, daß das Video-Ram auf einer bestimmten Adresse anfängt.

Nun ist aber nicht alles so schlimm, wie es auf den ersten Blick erscheint. Das Betriebssystem ist ja auch in der Lage, mit den wechselnden Startadressen klarzukommen, oder z.B. ein Zeichen aus der Bildschirm-Matrix zu bestimmen, wie das bei jeder Benutzung der Copy-Taste passiert. Die dafür benötigten Routinen können auch von selbsterstellten Maschinenprogrammen genutzt werden.
Viele dieser Routinen des Betriebssystems finden Sie in einem späteren Kapitel. Konkret zeigen wir die Nutzung der Grafik in einem Beispiel zum Zeichnen von Rechtecken und in einem Programm zum Erzeugen einer Grafik-Hardcopy.

1.7 Der Parallel-Schnittstellenbaustein 8255

Ursprünglich von INTEL für den 8080 entwickelt, eignet sich der 8255 als programmierbarer Mehrzweck-I/O-Baustein (I/O = Input/Output, Ein/Ausgabe) auch für andere Prozessoren. Der 8255 verfügt über insgesamt 24 Leitungen, über die Signale aus- oder eingegeben werden können. Jeweils 8 Leitungen bilden einen 8-Bit-Port, wobei der dritte Port in zwei getrennt programmierbare Hälften geteilt werden kann.

Die wichtigsten Leistungsmerkmale des 8255 sind:

24 programmierbare I/O-Anschlüsse.
Einfache Betriebsspannung 5 Volt.
Vollständig TTL-Kompatibel.
Drei leistungsfähige Betriebsarten programmierbar.
Jeder Port getrennt programmierbar.
Hoher Ausgangsstrom 1 mA bei 1.5 Volt Spannung.
Funktion Bit setzen/Bit Rücksetzen möglich.

1.7.1 Die Anschlußbelegung des 8255

Die Pinbelegung des 8255 ist im unten stehenden Bild gezeigt. Es bedeuten:

D0 - D7 : **Data Lines**. Diese Anschlüsse werden mit dem Datenbus des Prozessors verbunden. Sie dienen dem Transfer der Daten vom und zum Prozessor.

CS : **Chip Select**. Durch ein Low an diesem Anschluß wird der Baustein ausgewählt. Die jetzt an den RD-, WR- und Data-Leitungen anliegenden Signale werden vom 8255 akzeptiert.

RD : **Read**. Ein Low an diesem Anschluß veranlaßt den 8255 Daten oder Zustandsinformationen über den Datenbus an den Prozessor zu senden.

WR : **Write** wird Low, wenn der Prozessor Daten oder Steuerbefehle an den 8255 schicken will.

A0, A1 : **Adress Lines 0** und **1**. Über diese Anschlüsse wird zwischen den drei Datenkanälen und dem Steuer-Register ausgewählt. Häufig werden diese Anschlüsse mit den unteren beiden Adressleitungen des Prozessors verbunden.

PA 3	1	PA 4
PA 2		PA 5
PA 1		PA 6
PA 0		PA 7
RD*		WR*
CS*		RESET
GND		D 0
A 1		D 1
A 0		D 2
PC 7		D 3
PC 6		D 4
PC 5		D 5
PC 4		D 6
PC 0		D 7
PC 1		Vcc
PC 2		PB 7
PC 3		PB 6
PB 0		PB 5
PB 1		PB 4
PB 2		PB 3

1.7.1.1 Pinout des Parallelport 8255

RESET : Ein High an diesem Eingang setzt alle Register einschließlich des Steuerregisters zurück. Die Portleitungen werden in die Betriebsart Eingabe gebracht.

PA0 - PA7 : **Port A**. Diese acht Leitungen stellen den I/O-Port A dar und können wahlweise als Eingang oder Ausgang verwendet werden.

PB0 - PB7 : **Port B**. Funktion wie Port A.

PC0 - PC7 : **Port C**. Funktion wie Port A.

1.7.2 Die Betriebsarten des 8255

Bevor auf die vier internen Register eingegangen wird, müssen wir zunächst die Möglichkeiten des ICs etwas genauer betrachten. Wie schon zu Beginn erläutert, verfügt der 8255 über drei mögliche Betriebsarten:

Betriebsart 0 : Einfache Ein/Ausgabe
Betriebsart 1 : Getastete Ein/Ausgabe
Betriebsart 2 : Zweiweg-Bus

Die Betriebsart 0 ist die einfachste und auch häufigste. In diesem Mode kann bestimmt werden, ob die Ports als Ausgabe- oder als Eingabe-Leitungen arbeiten sollen. Werden Leitungen als Ausgang programmiert und wird auf diese Ausgänge vom Prozessor eine Information gelegt, so wird der Wert gespeichert und die Ausgänge bleiben bis zur Neuprogrammierung oder einem Reset erhalten.
Als Eingang programmierte Ports liefern beim Lesen den momentanen Zustand an diesen Leitungen.

Sowohl Port A wie auch Port B lassen sich nur als ganzer Port für die gewünschte Datenrichtung programmieren. Es ist also nicht möglich, z.B. die Portbits PA0, PA3 und PA7 als Ausgang und die verbleibenden Anschlüsse als Eingang zu verwenden.
Allerdings kann der Port C in zwei Hälften geteilt werden. Die Datenrichtung jeder Hälfte kann getrennt programmiert werden.

Die Betriebsart 1 unterscheidet sich grundsätzlich vom Mode 0. In dieser Betriebsart ist ein Datentransfer mit Hand-Shake-Signalen in einer Richtung möglich. Jetzt spricht man nicht mehr von drei vorhandenen Ports, die beiden Hälften des Port C werden den anderen beiden Ports als Steuer- und Quittungssignale zur Verfügung gestellt. Man spricht dann von den beiden Gruppen A und B.
Die Gruppe A besteht aus Port A und den Bits 4-7 des Port C, die Gruppe B entsprechend aus dem Port B und den Bits 0-3 des Port C.
Um die Progammierung des Mode 1 komfortabel zu gestalten, besteht die

Möglichkeit, jeweils ein spezielles Bit der entsprechenden Hälfte des Port B als Interrupt-Signal zu verwenden.
Ein solcher 8-Bit-Datentransfer wird z.B. bei Drucker-Schnittstellen verwendet. Hier zeigt ein Signal an, daß die Daten auf den Datenleitungen gültig sind. Ein rückgeführtes Signal meldet, ob der Empfänger, also in diesem Beispiel der Drucker, emfangsbereit ist oder ob die Daten korrekt emfangen wurden.
Diese Funktion kann vom 8255 wahlweise sowohl als Datenausgang wie auch als Eingang ausgeführt werden.

Die dritte Betriebsart (Mode 2) ist ein getasteter bidirektionaler Betrieb. Diese Funktion ist nur mit dem Port A möglich. Als Steuer- und Quittungssignale werden die Bits PC3-7 verwendet.
Ein möglicher Einsatz dieser Betriebsart wäre die Steuerung eines Floppy-Laufwerks, da hierbei die Daten ja sowohl zur Floppy wie auch von der Floppy zum Prozessor über dieselben Anschlüsse geführt werden müssen.

Zusätzlich besteht in allen drei Betriebsarten die Möglichkeit, die als Ausgang programmierten Bits des Port per Befehl gezielt zu setzen oder zu löschen.
Alle diese beschriebenen Betriebsarten lassen sich auch kombinieren. So ist es möglich, den Port A im Mode 0 als Ausgang, den Port B im Mode 1 als Eingang und die verbleibenden Bits des Port C als Eingang zu programmieren.

1.7.3 Steuerung des 8255, die Registerbeschreibung

Wenn man diese auf den ersten Blick verwirrende Anzahl der Möglichkeiten betrachtet, so fragt man sich unwillkürlich, wie alle Möglichkeiten und Kombinationen mit nur einem Steuerregister zu programmieren sind.
Der Trick, mit dem dies möglich wird, ist einfach. Das oberste Bit des Steuerworts wird als Kennzeichen-Bit verwendet. Ist dieses Bit im Steuerwort gesetzt, so haben die Bits 0 bis 6 die folgende Bedeutung:

Bit 0 : steuert Funktion **Port C** Bits 0 - 3
1 = Eingang
0 = Ausgang

Bit 1 : steuert Funktion **Port B**
1 = Eingang
0 = Ausgang

Bit 2 : wählt Mode **Gruppe B**
1 = Betriebsart 0
0 = Betriebsart 1

Bit 3 : steuert Funktion **Port C** Bits 4 - 7
1 = Eingang
0 = Ausgang

Bit 4 : steuert Funktion **Port A**
1 = Eingang
0 = Ausgang

Bit 6,5: wählen Modus **Gruppe A**
00 = Modus 0
01 = Modus 1
1x = Modus 2, Bit 5 ohne Bedeutung

Ist im Steuerwort das oberste Bit dagegen gelöscht, so wird über die Bits 0-3 die Funktion 'Bit setzen/Bit rücksetzen' des Port C definiert. Die Bedeutung dieser Bits lautet folgendermaßen:

Bit 0 : steuert Bit-Set/Bit-Reset
1 = Bit setzen
0 = Bit rücksetzen

Bits 3-1: Bitauswahl
000 = PC0
001 = PC1
010 = PC2
011 = PC3
100 = PC4
101 = PC5
110 = PC6
111 = PC7

Die Bits 4 bis 6 im Steuerwort sind bei gelöschtem siebtem Bit ohne Bedeutung.

Dieses Steuerregister kann nur beschrieben werden. Ein Lesen des Wertes ist nicht möglich. Wohl aber können die den Ports zugehörigen Register gelesen werden, auch wenn die Ports als Ausgang bestimmt sind. In diesem Fall entspricht der gelesene Wert dem Zustand der Portleitungen.

Der Zugriff auf die vier Register geschieht über die Anschluß-Pins A0 und A1. Diese Anschlüsse werden im 8255 decodiert und als Registerauswahlsignale benutzt. Üblicherweise liegen A0 und A1 des 8255 auf den gleichnamigen Adressleitungen des Prozessors. Dadurch ergibt sich dann eine durchgehende Adressierung über 4 Adressen.
Die Zuordung der Anschlüsse A0 und A1 zu den Registern zeigt die folgende Tabelle.

A1	A0	
0	0	Port A Register
0	1	Port B Register
1	0	Port C Register
1	1	Steuerregister

1.7.4 Der Einsatz des 8255 im CPC

Nachdem wir uns einen Überblick über die vielfältigen Einsatzmöglichkeiten des 8255 geschaffen haben, wollen wir uns mit dem praktischen Betrieb dieses universellen Schnittstellen-Bausteins im CPC auseinandersetzen. Wie eigentlich fast alle ICs im CPC 464 wird auch der 8255 optimal verwendet. Da bleibt kein Bit ungenutzt.

Doch werden wir konkret.
Der 8255 bedient die Tastatur, den Sound-Chip, den Motor des Cassetten-Recorders, erzeugt die Schreib-Signale des Recorders, liest den vom Recorder kommenden Bit-Strom, überprüft das V-Sync-Signal des CRTC, stellt fest, ob der Drucker empfangsbereit ist, fragt mit einem Bit den Zustand des EXP-Signals des Expansion Connectors ab, entscheidet über eine Brücke, ob die Erzeugung des Bildes nach PAL- oder SECAM-Norm mit 50 oder 60 Hertz Bildfrequenz erfolgen soll und zu guter Letzt bleiben noch ganze drei Bits über, die beim Einschalten Brücken abfragen und feststellen, was für einen Computer Sie sich gekauft haben. Der Zustand dieser Brücken entscheidet nämlich, ob Sie Schneider, Awa, Triumpf, Amstrad oder einen anderen der insgesamt acht verschiedenen möglichen Firmennamen in der Einschaltmeldung auf den Blidschirm bekommen.

All diese Funktionen mit den zur Verfügung stehenden 24 I/O-Leitungen zu realisieren, zeugt von der ausgesprochenen Sparsamkeit und Pfiffigkeit der Hardware-Entwickler.

Ein Blick auf den Schaltplan zeigt, wie der 8255 angeschlossen ist.
Der Datenbus ist direkt mit dem Datenbus des Prozessors verbunden. Das CS-Signal (Chip-Select) wird vom Adressbit A11 des Prozessors erzeugt. Die zur Registerauswahl vorhandenen Pins A0 und A1 des 8255 sind mit den Prozessoradresspins A8 und A9 verbunden.
Wie bereits erwähnt werden alle Peripherie-Bausteine im CPC über Port-Adressen angesprochen. Aus diesem Grund ist die Leitung RD* des 8255 mit dem Signal IORD* verbunden.
Erzeugt wird dieses Signal aus der Verknüpfung der Signale RD* und IORQ* des Z80 mit einem Gatter des IC112. Nur wenn IORQ* und RD* Low sind, erscheint am Ausgang Pin 6 des IC 74LS32 ein Low.
In ähnlicher Weise wird auch der WR*-Anschluß des 8255 angesteuert. Hier erscheint, vom Pin 3 des 74LS32 kommend, ein Low, wenn sowohl WR* als auch IORQ* des Z80 an den Pins 1 und 2 des IC112 Low werden.

Aus diesen Daten können nun die Port-Adressen des 8255 bestimmt werden. Um z.B. in das Register 0, das Datenregister des Port A, einen Wert zu schreiben, müssen die Anschlüsse A11, A9 und A8 Low sein. In binärer Schreibweise erhalten wir für das Highbyte des Adressbusses den Wert:

A15	A14	A13	A12	A11	A10	A09	A08
1	1	1	1	0	1	0	0

Das entspricht dem hexadezimalen Wert &F4.

Die unteren 8 Adressbits gehen in die Auswahl des 8255 nicht ein, hier ist jeder Wert zwischen &00 und &FF möglich.
Auch die gesetzten Bits im Highbyte sind zur korrekten Adressierung des 8255 eigentlich nicht nötig, und so könnte man auf die Idee kommen, als High-Byte den Wert 00H einzusetzen. Das würde sogar funktionieren. Da aber die Dekodierung der einzelnen Peripherie-ICs in ähnlich unvollständiger Weise vorgenommem wird, müssen die Bits gesetzt werden, sonst würden sich gleichzeitig andere ICs wie der CRTC oder das Gate Array mit angesprochen fühlen.

Doch zurück zu unserem Beispiel. Um also das Register A mit einem Wert zu laden, muß der Wert &F400 auf den Adressbus gelegt werden. Das kann mit den Befehlen

```
LD      A,wert
LD      BC,&F400
OUT     (C),A
```

erreicht werden. Entsprechend kann z.B. das Port-Register C mit den Befehlen

```
LD      BC,&F600
IN      A,(C)
```

ausgelesen werden.

Grundsätzlich werden alle drei Ports im Modus 0 verwendet. Somit stehen alle 24 Anschlüsse als I/O-Leitungen zur Verfügung.

Der **Port A (&F400)** ist mit den 8 Datenleitungen des Sound Generators AY-3-8912 verbunden. Je nach geforderter Aktion wird der Port A als Ausgang oder Eingang programmiert.
Als Ausgang programmiert werden über die 8 Portleitungen die Steuerbefehle an den Sound-Chip geschickt. Diese Steuerbefehle finden Sie detailliert im Kapitel über die Programmierung des AY-3-8912. An dieser

Stelle sei nur erwähnt, daß der Sound-Chip auch über einen bidirektionalen 8-Bit-Port verfügt. An diesen Port ist eine Seite der Tastatur-Matrix angeschlossen. Über den Port A des 8255 kann nun über den Umweg des Ports des AY-3-8912 festgestellt werden, ob eine Taste gedrückt ist. Zu diesem Zweck muß der Port A natürlich als Eingang programmiert werden.

Der **Port B (&F500)** wird fest als Eingangsport programmiert. Über diesen Port werden alle erwähnten Abfragen außer der Tastaturabfrage getätigt. Dabei sind die einzelnen Bits dieses Ports wie folgt belegt:

Bit 0 : Dieses Bit fragt den Zustand des V-Sync des CRTC ab. Da diese Abfrage recht rasch gehen muß, kann durch einfaches Rotieren des mit INP gelesenen Wertes das Bit 0 ins Carry-Flag geschoben werden. So kann schnell der Zustand des V-Sync festgestellt werden.

Bit 1-3 : Diese Bits entscheiden den Firmennamen in der Einschaltmeldung.

Bit 4 : Dieses Bit ist verbunden mit der Brücke LK4. Ist diese Brücke offen, so wird der Video-Controller für PAL-Betrieb mit 50 Hertz programmiert, eine geschlossene Brücke bewirkt eine Programmierung des CRTC für die SECAM-Norm mit 60 Hz Bildwiederholfrequenz. Diese unterschiedliche Programmierung ist wichtig, wenn der CPC 464 über das Modul MP1 an einem Fernseher betrieben werden soll.

Bit 5 : Dieses Bit fragt den Zustand des Signals EXP des Expansion Connectors ab.

Bit 6 : Dieses Bit gibt den Zustand eines angeschlossenen Druckers wieder. Da der Drucker nicht zu allen Zeiten Zeichen empfangen kann, besteht die Möglichkeit, durch High-legen dieses Anschlusses einen Zeichentransfer zu unterbinden.

Bit 7 : Über dieses Bit werden die vom Recorder mit TTL-Pegel gelieferten Daten eingelesen. Auch hier gilt das zu Bit 0 gesagte. Da diese Leitung sehr schnell geprüft werden muß, kann durch einmaliges Rotieren des Bit 7 in das Carry-Flag der Zustand dieser Leitung schnell bestimmt werden.

Der noch verbleibende **Port C (&F600)** ist im CPC fest als Ausgangsport programmiert. Mit vier seiner acht Leitungen steuert er einen Teil der

Tastaturabfrage und zwei weitere Bits werden für den Recorder verwendet. Die restlichen beiden Bits werden für die Ansteuerung des Sound-Chip benötigt. Da die Leitungen des Port C direkt gesetzt und gelöscht werden können, eignet er sich natürlich besonders für diese Aufgaben. Im einzelnen sind die Bits wie folgt verwendet:

Bit 0-3 : Diese Bits steuern die Tastaturmatrix. Die als Ausgang programmierten vier Leitungen sind verbunden mit dem IC101, einem BCD-Dezimal-Decoder.
Dieser Decoder legt entsprechend der binären Eingangsinformation einen seiner zehn Ausgänge auf Masse. Dabei sind als Eingangskombinationen die Werte zwischen 0 und 9 erlaubt.

Bit 4 : Dieses Bit steuert den Motor des Cassettenrecorders. Der Motor wird allerdings nicht direkt, sondern über einen Transistor (und ein nachgeschaltetes Relais) gesteuert. Liegt dieses Bit auf Masse, stoppt der Motor, bei einem High am Ausgang von PB4 leitet der Transistor Q101 und der Motor dreht sich bei gedrückter PLAY-Taste.

Bit 5 : Über diesen Pin des 8255 werden die Tonfrequenzen vom Computer geliefert, die auf dem Recorder aufgenommen werden sollen und beim Abhören diesen merkwürdigen Ton erzeugen.

Bit 6-7 : Diese Portbits sind mit den Anschlüssen BC1 und BDIR des Sound-Chip verbunden und arbeiten als Chip-Select- und Strobe-Signal für den AY-3-8912. Eine detailliertere Beschreibung dieser Anschlüsse finden Sie im nächsten Kapitel über den Sound-Generator.

1.8 Der programmierbare Sound Generator AY-3-8912

Der AY-3-8912 von General Instruments ist ein programmierbarer Sound Generator (PSG) der Spitzenklasse. Er wurde entwickelt für Telespiele, um diese mit besonders realistischem Sound zu versehen, nachdem die ersten Telespiele nur recht monotone Geräusche produzieren konnten. Um möglichst universell einsetzbar zu sein, wurde der PSG mit einer Vielzahl von Möglichkeiten zur Klangbeeinflussung versehen. Zusätzlich sagte man sich bei der Entwicklung dieses ICs wohl, daß in fast allen Einsatzgebieten auch irgendwelche Tasten, Joysticks oder Schalter abgefragt werden müssen. So gab man diesem PSG auch noch einen bidirektionalen 8-Bit-Parallelport mit.

Die Leistungsdaten dieses ICs im Überblick lauten:

Drei unabhängig programmierbare Ton-Oszillatoren
Ein programmierbarer Rausch-Generator
Vollständig software-gesteuerte Analogausgänge
Programmierbarer Mischer für Ton und Rauschen
15 logarithmisch gestufte Lautstärkestufen
Programmierbare Hüllkurven
Bidirektionaler 8-Bit-Datenport
TTL-Kompatibel
Einfache 5 Volt Betriebsspannung

Insgesamt verfügt der AY-3-8912 über 16 Register, von denen 15 Register genutzt werden können. Über diese Register können alle Klangmöglichkeiten des Chips programmiert werden.

Die Schaltung des PSG kann in einzelne Funktionsblöcke unterteilt werden.

Da ist zunächst der Block der Tongeneratoren. Die Tongeneratoren werden mit einem Taktsignal versorgt, das aus dem durch 16 geteilten Clock-Signal gewonnen wird. Die Tongeneratoren sind zuständig für die grundsätzliche Erzeugung der drei rechteckförmigen Tonfrequenzen.

Der Rauschgenerator erzeugt ein frequenzmoduliertes Rechtecksignal, dessen Pulsbreite von einem Pseudo-Rauschgenerator beeinflußt wird.

Die Mischer koppeln die Ausgangssignale der drei Generatoren mit dem Rauschsignal. Die Kopplung kann für jeden Kanal getrennt programmiert werden.

Der Funktionsblock der Amplitudenkontrolle bietet dem Anwender zwei Möglichkeiten. Zum einen kann die Ausgangsamplitude (die Lautstärke) der drei Kanäle über die Programmierung des entsprechenden Lautstär-

ke-Registers beeinflußt werden.
Alternativ besteht die Möglichkeit, sie vom PSG variabel zu beeinflussen. Dann wird der Ausgang des Hüllkurven-Registers genutzt, um die Lautstärke zu beeinflussen. Da die Hüllkurve (auch als Envelope, Umschlag oder Umhüllung bezeichnet) mit vier getrennten Parametern programmierbar ist, bestehen vielfältige Möglichkeiten der Tonbeeinflussung.

Der Funktionsblock der D/A-Wandler ist zuständig für die Erzeugung der Lautstärke der Ausgangssignale. Da die Lautstärke- und Envelope-Informationen als digitale Werte vorliegen, werden sie im D/A-Wandler umgesetzt.

Der letzte Funktionsblock hat mit der Tonerzeugung nichts zu tun. In diesem Block sind die zwei I/O-Ports untergebracht. Wenn Sie jetzt aufmerken, dann haben Sie aufmerksam gelesen. Tatsächlich enthält der Chip des AY-3-8912 zwei vollständige I/O-Ports, von denen aber nur einer an Anschlußpins herausgeführt ist. Derselbe Chip wird im AY-3-8910 eingesetzt, bei dem beide Ports zur Verfügung stehen.

1.8.1 Die Anschlüsse des Sound Chip

Da die Bezeichnungen der Anschlüsse des PSG nicht unbedingt selbsterklärend sind, hier die detaillierte Beschreibung der Funktion der Pins:

DA0 - 7 : Diese Anschlüsse des Sound Chips werden mit dem Datenbus des Prozessors verbunden. Die Bezeichnung DA deutet an, daß sowohl Daten als auch (Register-)Adressen über diese Anschlüsse geführt werden.

A8 : Dieser Anschluß kann als ein CHIP-SELECT-Signal verstanden werden. Um Register des PSG anzusprechen, muß dieser Anschluß High sein.

BDIR & BC1,2 : Der Anschluß BDIR-Signal (Bus DIRection) und die Anschlüsse BC1 und BC2 (Bus Control) steuern den Registerzugriff auf den PSG. Auf den ersten Blick mag die in der Tabelle gezeigte Zuordnung etwas seltsam erscheinen. Da dies IC jedoch ursprünglich als Baustein zum Prozessor 1610, einem speziellen 16-Bit-Prozessor von General Instruments, entwickelt wurde, hat man beim Entwurf auf die speziellen Eigenschaften und Steueranschlüsse dieses Prozessors Rücksicht genommen.

CHANNEL C	1		DA 0
TEST 1			DA 1
Vcc			DA 2
CHANNEL B			DA 3
CHANNEL A			DA 4
Vss			DA 5
IOA 7			DA 6
IOA 6			DA 7
IOA 5			BC 1
IOA 4			BC 2
IOA 3			BDIR
IOA 2			A 8
IOA 1			RESET*
IOA 0			CLOCK

1.8.1.1 Soundchip AY–3–8912

BDIR	BC2	BC1	Funktion des PSG
0	0	0	INACTIVE
0	0	1	LATCH ADRESS
0	1	0	INACTIVE
0	1	1	READ FROM PSG
1	0	0	LATCH ADRESS
1	0	1	INACTIVE
1	1	0	WRITE TO PSG
1	1	1	LATCH ADRESS

Innerhalb dieser Tabelle sind nur vier von acht Kombinationen wirklich sinnvoll. Darum wird häufig der Anschluß BC2 auf +5 Volt gelegt. Die verbleibende Tabelle wird nur noch von den Signalen BDIR und BC1 bestimmt und sieht folgendermaßen aus:

BDIR	BC1	Funktion
0	0	**INAKTIV**, der PSG-Datenbus ist hochohmig
0	1	**READ**, Daten können aus den PSG-Registern gelesen werden.
1	0	**WRITE**, Daten können in das gewählte PSG-Register geschrieben werden
1	1	**LATCH**, die Nummer oder Adresse des gewünschten PSG-Registers wird in den PSG geschrieben

ANALOG A : Dies ist der Ausgang des Kanals A. Hier können die von Kanal A erzeugten Töne abgenommen werden. Die maximale Ausgangsspannung ist 1 Vss.

ANALOG B : Funktion wie Pin 1 für den Kanal B.

ANALOG C : Funktion wie Pin 1 für den Kanal C.

IOA7 - 0 : Die IOA-Anschlüsse stellen den 8-Bit-Port des PSG dar. Je nach Programmierung arbeiten die Anschlüsse als Ausgang oder Eingang. Dabei kann nur die Betriebsart für den ganzen Port eingestellt werden. Ein gemischter Betrieb (gleichzeitig Bits als Eingang, andere Bits als Ausgang) ist nicht möglich.

CLOCK : Von der Frequenz dieses Signals werden alle Tonfrequenzen durch Teilung abgeleitet. Die Frequenz dieses Signals sollte zwischen 1 und 2 MHz liegen.

RESET : Durch einen Low-Pegel an diesem Anschluß werden alle internen Register zurückgesetzt. Ohne Reset stehen nach

dem Einschalten zufällige Werte in den verschiedenen Registern, die Folge wäre ein (wahrscheinlich) sehr unmusikalisches 'Geräusch'.

TEST1 : Test1 wird nur von der Herstellerfirma verwendet und muß im Betrieb unbeschaltet bleiben.

Vcc : An diesen Anschluß wird die Betriebsspannung von +5 Volt angelegt.

Vss : Dies ist der Masse-Anschluß des PSG.

1.8.2 Die Funktion der einzelnen Register des 8912

Da jetzt geklärt ist, wie über die Anschlüsse BDIR und BC1 die Register grundsätzlich angesprochen werden können, wollen wir sehen, was für Funktionen die Register ausführen. Dabei ist die Registernummer, die in der folgenden Aufstellung verwendet wird, gleich mit der Nummer, die im Adress-Register eingetragen werden muß, um das gewünschte Register anzusprechen.

Interessant ist noch die Tatsache, daß das Adressregister seinen Inhalt bis zur nächsten Programmierung behält. Man kann also ohne Probleme mehrmals nacheinander auf ein Datenregister zugreifen, ohne jedes Mal das Adressregister neu laden zu müssen.

Doch jetzt zur Registerbeschreibung.

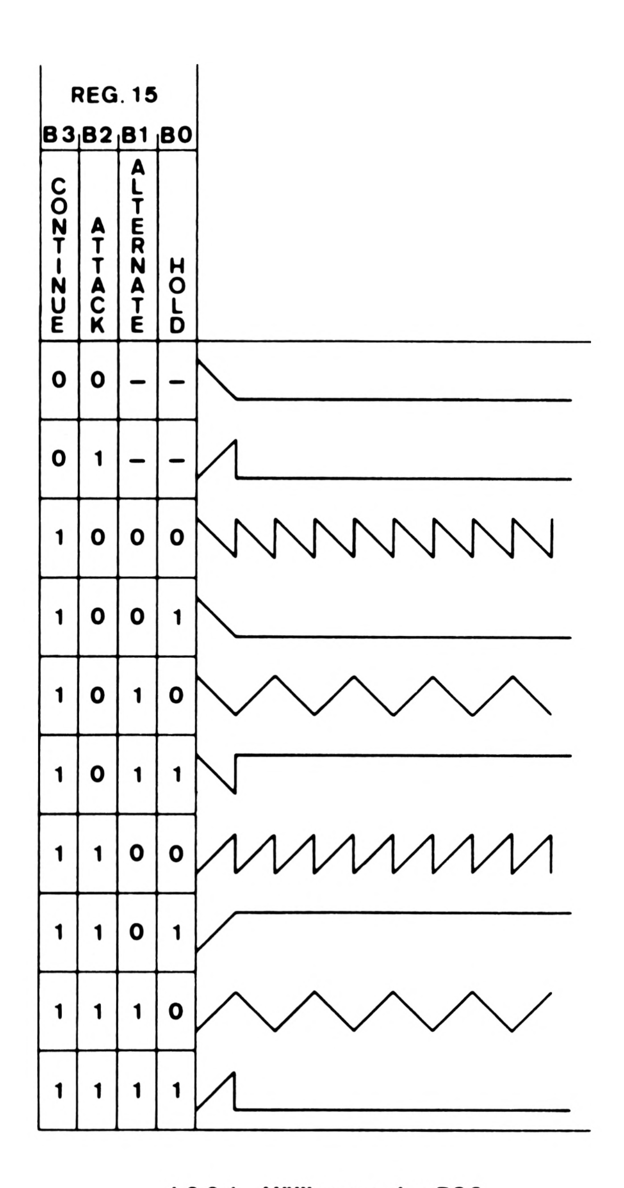
Reg 0,1 : Diese Register bestimmen die Periodendauer und damit die Frequenz des Tonsignals an ANALOG A. Allerdings sind nicht alle 16 Bit benutzt. Verwendung finden alle 8 Bit des Register 0 und die vier unteren Bit des Registers 1. Dabei kann die Frequenz mit dem Register 0 fein, mit dem Reg. 1 in groben Stufen beeinflußt werden. Je kleiner der 12-Bit-Wert dieser Register wird, desto höher wird der Ton.

Reg 2,3 : Funktion wie Reg 0,1, aber Kanal B.

Reg 4,5 : Funktion wie Reg 0,1, aber Kanal C.

Reg 6 : Dieses Register beeinflußt mit seinen unteren 5 Bit den Rausch-Generator. Auch hier gilt: je kleiner der Wert im Register, desto höher die Rausch-Frequenz.

Reg 7 : In diesem Multi-Funktionsregister kontrollieren die einzelnen Bits unterschiedliche Aufgaben. In der folgenden



1.8.2.1 Hüllkurven des PSG

Tabelle werden die Beeinflussungen genannt:

Bit 0	:	Ton von Kanal A ein-/ausschalten	0=ein / 1=aus
Bit 1	:	Ton von Kanal B ein-/ausschalten	0=ein / 1=aus
Bit 2	:	Ton von Kanal C ein-/ausschalten	0=ein / 1=aus
Bit 3	:	Rauschen zu Kanal A zu-/abschalten	0=zu / 1=ab
Bit 4	:	Rauschen zu Kanal B zu-/abschalten	0=zu / 1=ab
Bit 5	:	Rauschen zu Kanal C zu-/abschalten	0=zu / 1=ab
Bit 6	:	Port A als Ein-/Ausgang	0=in / 1=out
Bit 7	:	Port B als Ein-/Ausgang	0=in / 1=out

Reg 8 : Dieses Register bestimmt die Lautstärke des Signals an Kanal A. Zur Lautstärkeeinstellung werden die vier unteren Bits verwendet.
Das Bit 4 hat eine besondere Bedeutung. Ist es gesetzt, dann wird die Lautstärke durch das Hüllkurven-Register bestimmt, der Inhalt der Bits 0 bis 3 wird dann ignoriert.

Reg 9 : Wie Reg 8 für Kanal B.

Reg 10 : Wie Reg 8 für Kanal C.

Reg11,12 : Alle 16 Bit dieser beiden Register beeinflussen die Periodendauer der Hüllkurve. Der Inhalt des Reg. 11 wird als Low-Byte betrachtet, d.h. er beeinflußt die Periodendauer in feinen Schritten, das Reg. 12 ist das High-Byte des Hüllkurven-Generators.

Reg 13 : Die Bits 0 bis 3 dieses Registers bestimmen die Kurvenform des Hüllkurven-Generators. Die Zuordnung in Worten verständlich wiedergeben zu wollen, ist fast unmöglich. Die erzeugbaren Hüllkurven sind darum in der Grafik 1.8.2.1 gezeigt.

1.8.3 Der Betrieb des AY-3-8912 im CPC

In diesem Abschnitt wollen wir uns mit dem konkreten Anschluß und einigen mehr praktischen Dingen zum Betrieb des Sound Chip im CPC beschäftigen. Da die vorige Registerbeschreibung das Thema notgedrungen abstrakt und vielleicht nicht sehr anschaulich erläuterte, werden Sie nach Abschluß dieses Kapitels einige Spezialitäten des PSG besser verstehen.

Schauen wir zuerst einmal auf den Schaltplan.
Der PSG ist als IC102 zu finden.
Die Pins 3, 17 und 19 sind auf + 5 Volt gelegt. Über den Pin 3 bekommt der AY-3-8912 seine Betriebsspannung. Da BC2 (Pin 19) und A8 (Pin 17) auf + 5 Volt liegen, gehen sie in die Registerauswahl nicht ein.

Die verbleibenden Register-Steueranschlüsse BC1 (Pin 20) und BDIR (Pin 18) sind mit den Portbits PC6 und PC7 des 8255 verbunden. Je nach Zustand dieser Anschlüsse können dem PSG Registeradressen mitgeteilt sowie Daten in den PSG geschrieben oder herausgelesen werden.

Der eigentliche Adress- und Datentransfer geschieht über die PSG-Anschlüsse D0 bis D7, die mit dem Port A des 8255 verbunden sind. Je nach geforderter Aktion muß der Port A des 8255 als Ein- oder Ausgang programmiert werden.

Das Clock-Signal am Pin 15 ist ein Rechtecksignal mit einer Frequenz von 1 MHz. Dieses Signal wird durch Teilung der Quarzfrequenz vom Gate Array geliefert. Von diesem Signal werden durch Frequenzteilung alle Ton- und Hüllkurvenfrequenzen abgeleitet.

Der I/O-Port des PSG ist verbunden mit der Tastatur und dem Anschluß für den Joystick. Eine detaillierte Beschreibung der Tastatur und des Joysticks finden Sie in einem späteren Kapitel, hier sollen uns nur die klanglichen Möglichkeiten des Sound Chip interessieren.

Die wichtigsten Anschlüsse an diesem IC sind sicherlich die drei Analog-Ausgänge A, B und C an den Pins 1, 4 und 5. Diese Ausgänge sind als sogenannte Open-Emitter-Ausgänge ausgeführt. Um eine Tonwechselspannung ausgeben zu können, werden Widerstände benötigt, die zwischen Ausgang und Masse geschaltet werden. Diese Funktion haben die Widerstände R121, R122 und R123.
Von diesen Widerständen wird das Soundsignal einmal über die drei Widerstände R114, R115 und R116 zusammengemischt und steht als Mono-Signal am Anschluß 1 des Expansion Connectors zur Verfügung. Dieses Mono-Signal wird aber auch zum Steckverbinder CP001 geführt. Von hier gelangt das Signal zum internen Verstärker und Lautsprecher.
Zusätzlich werden die drei Ausgänge aber noch auf die Stereo-Klinkenbuchse an der Geräterückseite geführt. Dazu wird das Signal des Kanals B über die Widerstände R118/R119 gleichermaßen auf die beiden Stereo-Kanäle gelegt. Die Ausgänge A und C werden jeweils direkt über je einen Entkopplungskondensator (R117 und R120) auf einen der Stereo-Kanäle gelegt.
Durch diese Art der Beschaltung sind bei geschickter Programmierung sogar echte Stereoeffekte möglich. Denkbar wäre z.B. einen Ton zunächst nur über den Kanal A auszugeben. Nach einiger Zeit könnte derselbe Ton zusätzlich über den Kanal B ausgegeben werden. Dabei könnte die Lautstärke des Signals an Kanal B langsam steigen, die Lautstärke des Signals dagegen entsprechend reduziert werden. Durch diese Maßnahmen erscheint es, als würde der Ton von einer Ecke des Raums in die Mitte zwischen die beiden Lautsprecher-Boxen wandern. Von hier kann es jetzt bei Bedarf weiter in die andere Ecke gehen.
Diese Möglichkeiten sind sogar in Basic mit dem leistungsstarken

SOUND-Kommando möglich. Das Bedienungs-Handbuch ist aber bei der Angabe der Verteilung der drei Tonkanäle auf die zwei Stereokanäle widersprüchlich. Beachten Sie dies, wenn Sie Ihren CPC mit einer Stereoanlage verbinden. Nur Klänge des Kanals B erscheinen auf beiden Kanälen der Stereoanlage.

Wie aber erzeugt der PSG eigentlich die Töne? Betrachten wir einmal die Vorgänge an einem Kanal im Detail.
Wie schon erwähnt werden alle Töne vom Clock-Signal an Pin 15 abgeleitet. Zunächst wird das Taktsignal durch 16 geteilt. Daraus resultiert beim CPC eine Steuerfrequenz von 62,5 KHz. Diese Frequenz wird jetzt auf einen programmierbaren Frequenzteiler geführt. Je nach Inhalt der Tongenerator-Register wird die Steuerfrequenz weiter geteilt, um die gewünschte Tonfrequenz zu erhalten.
Dabei haben die Entwickler des ICs besonders tief in die Trickkiste gegriffen. Die Teilerkette besteht nicht nur aus Flip-Flops, die die Frequenz durch zwei teilen können. Durch eine spezielle Schaltungstechnik sind auch ungerade Teilerfaktoren möglich. Die Steuerfrequenz kann durchaus auch durch drei oder 17 geteilt werden. Dadurch erst können gerade im hohen Frequenzbereich alle benötigten Werte erzeugt werden.

Wenn Sie einmal im Handbuch des CPC im Anhang nachsehen, dann erhalten Sie für die Note D der vierten Oktave einen Periodenwert von 27. Wie kommt dieser Wert zustande?
Als wir uns diese Frage das erste Mal stellten, wurden Haare gerauft und Hände gerungen. Wie wir auch rechneten, es ergab sich kein vernünftiger Wert. Erst einige Stunden und mehrere Liter Kaffee später dämmerte die Erkenntnis, daß die so schön aufgemachte Tabelle im CPC-Handbuch falsch sein muß. Die Eingabe der Periode im SOUND-Kommando erzeugt eine Frequenz, die genau eine Oktave unter der angegebenen liegt. Die Eingabe von 'SOUND 1,284,100' erzeugt nicht die gewünschte Frequenz von 440 Hertz, produziert werden genau 220 Hertz!
Die für die Errechnung der Periode benötigte Formel muß richtig heißen:

PERIODE = ROUND (62500/FREQUENZ)

Scheinbar ist der Ersteller der Tabelle von einer Ansteuerfrequenz von 2 MHz ausgegangen.

Doch betrachten wir weiter die Erzeugung der Töne im PSG. Der Inhalt der Tongenerator-Register bestimmt also den Teilerfaktor für das Tonsignal. Wird das Register 0 des PSG mit dem Wert 100, das Register 1 mit dem Wert 0 geladen, so wird die Steuerfrequenz durch 100 geteilt. Am Ausgang der Teilerkette des Kanals A liegt ein Signal mit der Frequenz von 625 Hertz an.
Dieses Signal ist aber noch nicht am Ausgang A abzunehmen. Dazu muß der entsprechende Kanal zunächst eingeschaltet werden. Dies wird durch Löschen des entsprechenden Bits im Register 7 erreicht. Da wir in

unserem Beispiel den Kanal A gewählt haben, müssen wir das Bit 0 löschen. Dabei ist der Zustand der übrigen Bits zu beachten. Beim CPC bedeutet dies konkret, das Bit 6 nicht ungewollt zu verändern, da sonst die Tastatur gesperrt wird.
Aber auch jetzt ist wahrscheinlich noch kein Ton zu hören, da noch die Lautstärke des jeweiligen Kanals einzustellen ist. Für Kanal A ist das Register 8 zuständig. Ein Wert von 1 erzeugt nun einen leisen Ton, bei einem Wert von 15 erhalten wir die maximale Lautstärke.

Setzen wir das Bit 4 im Lautstärke-Register, dann wird die Information in den Bits 0 bis 3 ignoriert. Jetzt bestimmen die Register 11, 12 und 13 die Lautstärke. Allerdings ist die Lautstärke jetzt nicht mehr auf einen Wert fixiert, sondern variabel.

Betrachten wir zunächst das Register 13. Dieses Register trägt den offiziellen Namen **'ENVELOPE SHAPE/CYCLE CONTROL REGISTER'**. Die Funktion wird am besten in einem kleinen Beispiel deutlich.
Nachdem wir die Register 0, 1, 7 und 8 mit den entsprechenden Werten versorgt haben, schreiben wir einmal in das Reg. 13 den Wert 12. Jetzt sind die Bits 2 und 3 gesetzt, die unteren 2 Bits gelöscht.
Die in der Registerbeschreibung gezeigte Tabelle zeigt bei dieser Kombination eine Folge langsam ansteigender und schnell abfallender 'Zakken'. In der Praxis bedeutet dies, daß die Lautstärke des Tons zunächst langsam bis zum Maximum ansteigt. Dann wird der Ton abgeschaltet und die Lautstärke nimmt wieder zu. Dieser Zustand bleibt erhalten, bis ein neues Kommando zum Register 13 geschickt wird.

Die Zeit des Ansteigens der Lautstärke kann über die Register 11 und 12 eingestellt werden. Diese Register beeinflussen ähnlich den Tongenerator-Registern eine weitere programmierbare Teilerkette im PSG. Die Teilerkette wird mit einem Signal versorgt, das dem durch 256 geteilten Clock-Signal entspricht. Das ergibt eine Frequenz von 3906.25 Hertz entsprechend einer Periodendauer von etwa 250 Microsekunden.

Wird in das Reg. 11 ein Wert 1, in das als High-Byte arbeitende Reg 12 der Wert 0 geschrieben, so wird tatsächlich die Lautstärke des Tones in 250 Microsekunden von 0 bis zur maximalen Lautstärke hochgeregelt. Das liegt aber schon im Bereich der hörbaren Töne und erzeugt einen deutlichen Pfeifton, der dem eigentlich gewünschten Ton überlagert wird.
Aus diesem Grund werden die Registerwerte immer deutlich größer gewählt werden. Beim Maximalwert (255 ins Reg 11 und Reg 12) dauert das Ansteigen bis zur vollen Lautstärke ganze 16,8 Sekunden.

Diese Beeinflussung der Lautstärke über die Envelope-Register wird von der Software des CPC nicht verwendet. Das ENV-Kommando beeinflußt die Lautstärke des Tons nur über Manipulationen der unteren vier Bit des Lautstärke-Registers. Das ENT-Kommando des CPC hat im PSG überhaupt kein Equivalent. Diese Funktion wird durch geschicktes Verändern der Tongenerator-Register erzeugt.

1.9 Die Schnittstellen des CPC 464

Der Begriff Schnittstelle läßt sich definieren als Verbindungsstelle zwischen Computer und Außenwelt. Dabei kann die Außenwelt sowohl ein anderer Computer, ein Drucker oder sonstige Peripherie, ein Meßgerät oder auch der Mensch sein. Nach dieser Definition von Außenwelt wollen wir in diesem Kapitel nicht nur die an der Geräte-Rückseite angebrachten Steckverbindungen beschreiben, sondern auch Tastatur, Monitoranschluß und Recorder mit einbeziehen.
Die für den Benutzer wichtigsten Schnittstellen sind Tastatur und Monitor, da diese den unmittelbaren Kontakt zum Computer darstellen. Fangen wir darum mit diesen beiden an.

1.9.1 Die Tastatur

Insgesamt sind auf der CPC-Tastatur 74 Tasten untergebracht. Da die beiden SHIFT-Tasten parallel geschaltet sind, sind also 73 einzelne Tasten abzufragen.
Die Matrix, in der die Tasten angeordnet sind, besteht aus 8 mal 10 Leitungen. Da auch die Joysticks über diese Matrix abgefragt werden, werden insgesamt 79 Tastenpositionen belegt. Der zweite Joystick, über die Buchse im ersten angeschlossen wird, aber nicht auf eigene Positionen der Matrix geführt, die zugehörigen Schalter sind zu Tasten der Tastatur parallel geschaltet.

Hardwaremäßig wird die Tastatur über den 8255 und den Sound Chip abgefragt. Das funktioniert im einzelnen etwa folgendermaßen.
Der 8255 liefert an den Portausgängen PC0 bis PC3 ein Halbbyte, das durch den Decoder IC101 in eine dezimale Information gewandelt wird. Je nach anliegender Eingangsinformation wird einer der zehn Ausgänge Low. Dieser Decoder, ein 74LS145 wird darum auch BCD-Dezimal-Decoder genannt. Liegt die Eingangsinformation nicht im Bereich von 0 bis 9, dann liegen alle Decoder-Ausgänge auf High.

Der Parallel-Port des Sound Chip ist für die Tastatur-Abfrage als Eingangsport programmiert. Liegt an diesen Eingängen kein Signal an, dann erhält man beim Lesen des Ports an allen Eingängen eine 1, insgesamt also &FF.

Es sei jetzt einmal die Eingangsinformation des Decoders &04. Entsprechend wird der Ausgang Pin 5 Low. Davon nimmt der Port des Sound Chip aber so lange keine Notiz, wie keine entsprechende Taste gedrückt wird. Ein Druck auf die ESC-Taste z.B. hat zu diesem Zeitpunkt keine Auswirkung, da der Ausgang Pin 8 des Decoders High ist. Wird aber die SPACE-Taste gedrückt, dann ändert sich der vom Sound Chip gelieferte Wert. Jetzt liegt durch die gedrückte Taste das Bit 7 des Port an Masse und wir erhalten den Wert &7F vom Sound Chip.

Keyboard Connector

Lines	3	4	5	6	7	8	9	10
above 2								DEL
2–11	! 1	" 2	ESC	Q	TAB	A	CAPS LOOK	Z
11–12	$ 4	# 3	E	W	S	D	C	X
12–13	& 6	% 5	R	T	G	F	B	V
13–14	(8	' 7	U	Y	H	J	N	SPACE
14–15	_ 0	) 9	O	I	L	K	M	< ,
15–16	£ ↑	– =	| @	P	+ ;	* :	? /	> .
16–17	CLR	[	ENTER	]	* 4	SHIFT	\	CTRL
17–18	←	COPY	7	* 8	* 5	* 1	* 2	* 0
18–19	↑	→	↓	* 9	* 6	* 3	* ENTER	* .

1.9.1.1 Die Tastaturmatrix

Fünfzigmal in der Sekunde werden alle Tasten einmal überprüft. Dazu werden nacheinander an die vier verwendeten Ausgänge des Port C die Werte 0 bis 9 ausgegeben und nach jeder Ausgabe der Wert des Sound Chip geprüft. Werden dabei irgendwelche gedrückten Tasten registriert, so werden die gedrückten Tasten in einer Tabelle gespeichert und bei Bedarf in Tastennummern und die entsprechenden Zeichen umgerechnet.

Sehr angenehm an der Tastatur ist die Tatsache, daß bis zu 20 Zeichen zwischengespeichert werden. In Basic-Programmen kann man schon Eingaben machen, während der Computer noch Berechnungen vornimmt oder mit der Bildschirmausgabe beschäftigt ist. Nur bei Benutzung des Recorders ist die Tastaturabfrage gesperrt, da die dafür benötigte Zeit auf Grund des kritischen Timings nicht zur Verfügung steht. Einzige Ausnahme ist die ESC-Taste, die ja möglicherweise zum Abbruch der Cassettenoperation benötigt wird.

Übrigens gibt es bei der Tastatur eine kleine Besonderheit. Drücken Sie doch einmal gleichzeitig die Tasten J, K und L. Zur großen Überraschung erscheint auch noch ein H auf dem Bildschirm. Dies passiert immer, wenn drei Tasten gedrückt werden, die die Ecken eines Vierecks in der Tastatur-Matrix bilden, also auch bei 123 oder DFG. In diesem Fall erscheint gleichzeitig das vierte Zeichen der Matrix.
Dieser 'Fehler' ist nicht weiter schwerwiegend, allerdings können Programme auch durch gleichzeitiges Drücken der Tasten 2,3 und E beendet werden.

1.9.2 Der Video-Anschluß

Der Video-Anschluß des CPC stellt alle Signale für den Betrieb eines Monitors zur Verfügung. Dabei ist es unerheblich, ob es sich um den mitgelieferten Monitor oder einen (fast) beliebigen anderen handelt.

Das Gate Array liefert für den Monitor vier Signale. Drei Signale enthalten die Information über die Farbe, das vierte Signal ist eine Mischung aus den CRTC-Signalen V-Sync und H-Sync.
Diese Signale werden mit den Widerständen R131, R 132 und R133 sowie R195 gemischt und mit dem Transistor Q102 verstärkt. Das entstehende Ausgangssignal hat die Bezeichnung LUM und dient den grünen Monitoren als Video-Signal. Aber auch handelsübliche Farbmonitore mit einfachem Video-Eingang können über dieses Signal bei Darstellung aller Farben betrieben werden.

1.9.3 Der Recorder

...ist die Cassette ein ausgezeichneter Standard-Datenspeicher zu einem sehr günstigen Preis.

Dieses Zitat aus dem CPC-Bedienungshandbuch (Kapitel 2 Seite 9) ist uneingeschränkt richtig.

Auch wenn Sie ein Floppy-Laufwerk besitzen oder später anschaffen wollen, so wird der eingebaute Recorder sicher auch weiterhin gute Dienste leisten. Da die vom CPC verwendeten 3-Zoll-Disketten noch relativ teuer sind, kann die Cassette als sehr preiswertes 'Back-Up-Medium' genutzt werden.

Der Recorder selbst ist ein handelsübliches Laufwerk, wie es auch in einfachen Cassetten- und Radio-Recordern zu finden ist. Das erklärt auch das Vorhandensein der eigentlich sinnlosen Pause-Taste.

Die Elektronik des Recorders ist allerdings den Bedürfnissen im CPC angepaßt worden. Das Ausgangssignal ist ein Rechteck-Signal mit annähernd 5 Volt Amplitude. Dadurch kann es direkt vom Bit 7 des Port B des 8255 verarbeitet werden.
Auch der Audio-Verstärker, über den der Sound des CPC zu hören ist, wurde auf der Recorder-Platine untergebracht.

Doch wenden wir uns dem verwendeten Aufzeichnungsformat zu.
Grundsätzlich kann der Recorder die Daten nur bitweise speichern. Jedes zu speichernde Byte muß also in die einzelnen Bits zerlegt und übertragen werden. Diese Zerlegung wird vom Prozessor per Software vorgenommen, wobei zuerst das höchstwertige Bit zum Recorder geschickt wird.

Das vom 8255 gelieferte Signal für den Recorder ist ein Rechtecksignal. Jedes Bit wird als eine Rechteck-Schwingung aufgezeichnet, bei der die Low-Phase genau so lang ist wie die High-Phase. Man sagt auch, das Rechteck-Signal habe ein Tastverhältnis von 1:1. Ein 0-Bit benötigt die halbe Zeit eines 1-Bits.
Aus diesem Grund sind die Angaben über die Aufzeichnungsgeschwindigkeit auch nur ungefähre Angaben. Es ist offensichtlich, daß ein Datenblock aus lauter 0-Bytes in der Hälfte der Zeit gespeichert und geladen werden kann, wie ein ebensolanger Block, der ausschließlich aus &FF besteht. Da aber statistisch die Verteilung von 0- und 1-Bits in einem Datenblock etwa gleich ist, kann man von den Angaben 1000 Baud (1 Baud = 1 Bit pro Sekunde) bei SUPER-SAVE und 2000 Baud bei SPEED-LOAD ausgehen.

Jede Datei, unabhängig, ob es sich um Programme oder Daten handelt,

kann maximal 65536 Bytes lang sein. Die Dateien werden in Blocks übertragen, die jeweils 2048 Bytes enthalten. Jeder Block enthält maximal acht 256 Bytes große Datensegmente. Vor jedem Block wird ein Header, also ein Kopf oder Vorspann, übertragen.

Obwohl es keine elektrische Verbindung zum Verstärker und Lautsprecher gibt, kann trotzdem auch bei niedrig eingestellter Lautstärke das Laden und Abspeichern von Daten und Programmen verfolgt werden.

Der Header der Blocks ist akustisch einfach zu identifizieren. Es ist der zu Beginn eines jeden Blocks hörbare lange gleichmäßige Ton sowie einige folgende Bytes, die aber mit dem Ohr nicht zu unterscheiden sind.
Der lange, gleichmäßige Ton ist eine Serie von 2048 1-Bits. Nach diesen Bits folgt ein einziges 0-Bit und darauf ein Synchronisationsbyte. Die lange Folge der 1-Bits zu Beginn wird vom Rechner benötigt, um die Aufnahme-Baud-Rate zu bestimmen. Das 0-Bit zeigt dem Rechner, daß dieser Vorspann beendet ist und das Sync-Byte wird benötigt, um zwischen der Header-Information und den Daten zu unterscheiden.
Die Header-Information steht in einem 64 Byte langen Datenbereich, der vor jedem 2K-Daten-Block übertragen wird. In diesem Header-File finden sich Informationen über die eigentliche Datei, z.B. der Name, ob das File geschützt ist oder nicht, ob es sich um ein Basic-Programm oder eine Ascii-Datei handelt und wie lang das Programm ist.

Der genaue Aufbau dieses Header ist folgendermaßen:

Bytes 0-15 : **Name der Datei**, wenn kürzer als 16 Bytes, dann mit 00 aufgefüllt.

Byte 16 : **Block-Nummer**, in diesem Byte steht die Nummer, die beim Laden oder auch beim Catalog angezeigt wird.

Byte 17 : Steht in diesem Byte ein anderer Wert als 00, dann handelt es sich um den **letzten Block der Datei.**

Byte 18 : Dieses Byte enthält den **File-Typen**. Die Information ist in den einzelnen Bits verschlüsselt. Die Bedeutung der Bits folgt im Anschluß an diese Tabelle.

Bytes 19,20 : In diesen Bytes ist die **Länge der File-Information dieses Blocks** enthalten. Ist der Block, also die 2 K, voll beschrieben, so enthalten diese Bytes den Wert &0800, beim letzten Block oder bei Programmen, die kürzer als 2 K sind, ist hier die Anzahl der Bytes des Blocks enthalten.

Bytes 21,22 : Diese Bytes geben die **Ladeadresse** an, von wo die Daten ursprünglich geschrieben wurden. Bei Basic-Programmen

ist das die Adresse 368 Dezimal, bei Binär-Files, also Maschinensprache, normalerweise die Adresse, an der das Programm im Speicher läuft.

Byte 23 : Ist der Inhalt dieses Bytes ungleich Null, dann handelt es sich bei dem Block um den **ersten Block des Files**.

Bytes 24,25 : In diesen Bytes ist die **Länge des Files** enthalten.

Bytes 26,27 : Die Möglichkeiten dieser Bytes werden leider nicht direkt vom Basic des CPC unterstützt. Sie enthalten die **Startadresse eines Maschinensprache-Files**, die ja nicht unbedingt mit der Ladeadresse übereinstimmen muß. Diese Bytes ermöglichen bei entsprechender Programmierung des Ladeprogramms einen 'Auto-Start'.

Die restlichen Bytes 28 bis 63 des Header werden nicht vom Betriebssystem genutzt und stehen dem versierten Programmierer zur Verfügung.

Doch jetzt die Aufschlüsselung der Bits im Byte 18 des Header.

Bit 0 : Ist dieses Bit gesetzt, so ist das entsprechende File als geschützt erklärt. Geschützte Programme können von Basic aus mit 'SAVE "NAME",p' erzeugt werden.

Bit 1-3 : Diese Bits bestimmen den Typ des Files. Obwohl mit drei Bit acht verschiedene File-Typen möglich sind, werden nur die File-Typen Basic-Prg (0), Binärfile (1) und Asciidatei (3) verwendet.

Bit 4-7 : In diesen Bits ist normalerweise eine 0 zu finden, nur Asciidateien haben im Bit 4 eine 1.

Wie bereits erwähnt wird die gespeicherte Information in den einzelnen Blocks weiter unterteilt zu einzelnen Segmenten. Jedes Segment besteht aus 256 Daten-Bytes und Checksummen-Bytes. Die Checksumme jedes Segments wird nach einer speziellen Formel berechnet und erlaubt es, beim Lesen des Files zu prüfen, ob die Bits ordnungsgemäß übertragen wurden. Sobald die errechnete Checksumme nicht mit den gelesenen Werten übereinstimmt, wird der READ ERROR B angezeigt.
Der READ ERROR A zeigt an, daß ein Bit gelesen wurde, dessen Zeit zu lang für die errechneten Werte für Null- oder Eins-Bits ist. Dieser Fehler entsteht häufig beim Lesen von Programmen, wenn bei der Aufnahme die Cassette klemmte und jetzt bei der Wiedergabe 'leiert'.
Der dritte mögliche Fehler ist der READ ERROR D. Dieser Fehler dürfte nur in den seltensten Fällen auftreten, da er signalisiert, daß der gelesene Block länger als die zulässigen 2048 Bytes ist. Das kann aber nur auftre-

ten, wenn der Anwender beim Speichern in die Header-Information größere Werte als erlaubt einträgt.

Sicher kennen Sie den Basic-Befehl **'SPEED WRITE par'**. Je nach verwendetem Parameter werden Daten mit durchschnittlich 1000 oder 2000 Baud auf Cassette gespeichert. Damit ist aber noch nicht die obere Grenze der Geschwindigkeit erreicht. Durch Verwendung einer Betriebssystem-Routine läßt sich jede Baudrate zwischen 700 Baud und etwa 3600 Baud einstellen. Die benötigte Routine hat ihren Einsprung auf der Adresse &BC68. Sie erwartet in zwei Registern Parameter und stellt entsprechend die Schreibgeschwindigkeit ein. Ein Wert wird im HL-Registerpaar übergeben und bestimmt die Baudrate. Die Formel zur Bestimmung dieses Wertes lautet:

Baudrate = 333333/halbe Länge eines Null-Bits

Bei 1000 Baud ergibt sich daraus eine Zeit von 666 Microsekunden für ein Null-Bit, ein Eins-Bit ist genau doppelt so lang.

Die im Recorder verwendete Elektronik hat aber eine Besonderheit. Werden abwechselnd Null- und Eins-Bits gelesen, so versucht die Elektronik die Zeitunterschiede auszugleichen. Dadurch werden Eins-Bits kürzer, Null-Bits aber erscheinen als längere Impulse als nach der Aufzeichnung zu erwarten wären. Aus diesem Grund muß eine Vorkompensation durchgeführt werden, die Null-Bits werden kürzer aufgezeichnet, Eins-Bits werden mit geringfügig längeren Zeiten aufgezeichnet. Diese für die Vorkompensation benötigten Zeiten werden im Akku der Routine übergeben.

Für Versuche zur Bestimmung der höchsten, halbwegs zuverlässigen Schreibgeschwindigkeit genügt es, im Akku einen Wert von 10 zu übergeben. Um mit 3600 Baud aufzuzeichnen, muß die folgende Routine einmal aktiviert werden:

```
LD HL,93
LD A,10
CALL &BC68
RET
```

Diese wenigen Bytes können leicht mit den folgenden Zeilen in den Speicher gelegt werden:

```
10  MEMORY HIMEM - 10
20  FOR I = 1 TO 9
30  READ X  :  POKE HIMEM + I,X
40  NEXT I
50  CALL HIMEM + 1
60  DATA &21,&5D,&00,&3E,&0A,&CD,&68,&BC,&C9
```

Varieren Sie ruhig etwas mit den Werten in HL und Akku (der zweite und der fünfte Wert in der Data-Zeile), um die maximale Aufzeichnungsfrequenz zu bestimmen. Diese ist vom verwendeten Cassettenmaterial abhängig. Aber auch die Gleichlaufeigenschaften Ihres Recorders spielen eine nicht unerhebliche Rolle bei der Zuverlässigkeit hoher Aufnahmegeschwindigkeiten.

Werden die Werte zu klein gewählt, dann kann der CPC die geforderten Zeiten nicht mehr einhalten, als Ergebnis erhalten Sie die Fehlermeldung WRITE ERROR A.

Zum Schluß noch ein Tip:
Sie werden sicher beim Abspeichern sehr langer Programme mit vielen Variablen bemerkt haben, daß es bis zu 15 Minuten dauern kann, bis die Daten oder das Programm gespeichert sind. Das liegt an der Tatsache, daß der CPC zum Speichern einen Bereich von 2K für die zu übertragenden Blocks benötigt. Dieser Buffer wird an der oberen Speichergrenze angelegt. Ist dieser Bereich jedoch mit Variablen belegt, dann werden diese Variablen in einen anderen Speicherbereich copiert. Dieser Vorgang ist vergleichbar mit der viel gefürchteten Garbage Collection, die immer dann auftritt, wenn im Speicher für Zeichenketten und Arrays kein ausreichender Platz vorhanden ist.
Die durch die Variablenverschiebung auftretende Wartezeit kann man aber deutlich reduzieren, indem zu Beginn des jeweiligen Programms dieser 2k-Puffer bereits angelegt und geschützt wird. Ein möglicher Programmanfang könnte folgendermaßen aussehen:

```
10  OPENOUT "DUMMY"
20  MEMORY HIMEM - 1
30  CLOSEOUT
40
50  'REST DES PROGRAMMS
```

Dieser Vorgang ist natürlich nur sinnvoll, wenn Sie in dem entsprechenden Programm auch mit Dateien arbeiten. Ist das nicht der Fall, dann können Sie auf die gezeigten Programmzeilen verzichten und vor dem gewünschten Abspeichern den Befehl CLEAR eingeben. Dadurch werden alle zuvor definierten Variablen gelöscht und das Anlegen des Cassettenpuffers geht ohne nennenswerte Zeit vonstatten.

1.9.4 Die Centronics-Druckerschnittstelle

Man findet an jedem Computer etwas, das man für verbesserungswürdig hält. Beim CPC ist das ohne Frage die Druckerschnittstelle. Hier ist leider zu stark gespart worden.
Wir meinen nicht die mechanische Ausführung der Steckverbindung. Die getroffene Wahl ist sicherlich die für den Hersteller preiswerteste, aber auch für den Computerbesitzer nicht von Nachteil, da die benötigten Steckverbinder mittlerweile recht preiswert zu haben und auch recht zuverlässig sind.
Ursache für unseren 'Unmut' ist die Tatsache, daß die Schnittstelle über nur sieben Datenbits verfügt. Die meisten Drucker, sogar der von Schneider zum CPC angebotene, haben einen 8-Bit-Eingang, und entsprechend sind viele Kommandos und Möglichkeiten dieser Drucker nur über Umwege oder überhaupt nicht zu erreichen.

Aber betrachten wir zunächst den hardwaremäßigen Aufbau der Schnittstelle.
In der Hauptsache besteht die Schnittstelle aus dem IC106, einem 8-fachen Latch 74LS273. Die acht einzelnen Latches arbeiten wie Flip-Flops, die an den Eingängen liegende Information wird mit einer High-Low-Flanke am Takt-Eingang Pin 11 gespeichert und steht bis zu einem RESET oder einer Neuprogrammierung an den Ausgängen zur Verfügung, unabhängig von sich ändernden Eingangssignalen.
Das Taktsignal, dessen High-Low-Flanke das Speichern der Eingangswerte bewirkt, wird mit dem OR-Gatter 74LS32, IC112, Pins 11, 12 und 13 erzeugt. Der Ausgang Pin 11 wird dann Low, wenn beide Eingänge Low sind.
Auch der Druckeranschluß wird über die Portadressierung angesprochen. Aus diesem Grund liegt das Signal IOWR* an einem Eingang des OR-Gatters, am anderen Eingang liegt die Adressleitung A12.
Wie auch bei den anderen Peripherie-Bausteinen ist die Dekodierung also sehr unvollständig. Entsprechend müssen alle Adressleitungen, die nicht für die Decodierung benötigt werden, High sein, um Kollisionen mit anderen verwendeten Portadressen zu vermeiden. Damit ergibt sich eine effektive Portadresse von &EFxx.

Die Eingänge des Drucker-Latch sind mit dem Prozessor-Datenbus verbunden. Die Ausgänge liegen am Druckeranschluß. Nur das Bit 7 wird über ein als Inverter benutztes NAND-Gatter des IC110 an den Centronics-Port gelegt. Dies Bit stellt das für den Drucker benötigte Strobe-Signal dar. Normalerweise ist dies Signal High. Will der Rechner aber ein Zeichen an den Drucker schicken, so legt er das zu übertragende Byte auf die Datenleitungen und kurz darauf das Strobe-Signal auf Low. Damit wird das zu übertragende Byte von Drucker akzeptiert.
Voraussetzung dafür ist allerdings, daß das Signal Busy des Druckers Low ist. Der Zustand des Busy-Signals wird vom Bit 6 des 8255-Ports B abgefragt.

Wie aber kann das Strobe-Signal erzeugt werden? Nichts einfacher als das.
Jedes zu übertragende Byte wird zuerst mit &7F verUNDet. Damit ist das oberste Bit des Bytes mit Sicherheit gelöscht. Dieses Byte wird per OUT-Befehl auf den Printer-Port ausgegeben.
Jetzt liegen die zu übertragenden Bits bereits am Drucker an, das Strobe-Signal ist über den Inverter aber immer noch High. Darum wird anschließend mit OR &80 das Bit 7 des auszugebenden Wertes gesetzt und ebenfalls auf den Printer-Port ausgegeben. An dem zu übertragenden Wert hat sich nichts geändert, nur das Strobe-Signal ist durch den Inverter Low geworden.
Dieses Signal muß aber auch wieder High werden, darum wird mit UND das oberste Bit wieder gelöscht und das Byte noch einmal ausgegeben. Damit ist ein Byte vom Rechner zum Drucker geschickt worden.

Von Basic aus ist die Ausgabe auf den Drucker kein Problem. Aber auch in Maschinensprache muß nicht der ganze 'Kram' selbst geschrieben werden. Es gibt mehrere Routinen, die einem einigen Programmieraufwand abnehmen.
Da ist zunächst die Routine mit Einsprung bei &BD2B. Über diese Routine kann man ein Zeichen auf den Drucker ausgeben. Das jeweilige Zeichen muß sich dabei im Akku befinden. Zusätzlich prüft diese Routine, ob der Drucker 'Busy' ist. Meldet sich der Drucker innerhalb von 0.4 Sekunden nicht, dann kehrt die Routine mit gelöschtem Carry-Flag zurück. Dann muß ein neuer Versuch mit demselben Zeichen gestartet werden. Diese Routine wird auch vom Basic-Interpreter verwendet. Bei geglückter Übertragung ist das Carry gesetzt. Daraufhin kann das nächste Zeichen gesendet werden.

Eine weitere Routine hat ihren Einsprung drei Bytes weiter (&BD2E). Diese Routine kann genutzt werden, um den Zustand des Druckers zu prüfen. Ist kein Drucker angeschlossen oder meldet der Drucker 'Busy', kann also momentan keine Zeichen entgegen nehmen, dann kehrt diese Routine mit gesetztem Carry zurück, andernfalls ist das Carry gelöscht.

Die dritte verwertbare Routine (&BD31) erledigt alle Vorgänge, um ein Zeichen auf dem Drucker auszugeben. Dabei muß aber der Programmierer vorher prüfen, ob der Drucker empfangsbereit ist und das gewünschte Zeichen im Akku übergeben. Wird die Überprüfung des Zustands versäumt, dann geht das Zeichen eventuell ins 'Leere'.

Wie diese Routinen einsetzbar sind, finden Sie etwas später in diesem Buch. Dort zeigen wir am Beispiel einer Text- und einer Grafik-Hardcopy den Einsatz dieser und anderer Routinen, die den Gebrauch verdeutlicht.

Aber noch eine Besonderheit gilt es bei der vorhandenen Beschaltung des Centronics-Anschlusses zu berücksichtigen.

Die Kontaktbelegung des Druckerports verleitet geradezu, sich die benötigten Steckverbindungen und ein Stück Flachbandkabel zu besorgen und sich ein solches Kabel selbst zu fertigen. Handelt es sich bei den Verbindern dazu noch um sogenannte Quetschverbinder, dann haben auch handwerklich ungeschickte CPC-Besitzer ein solches Kabel in 5 bis 10 Minuten selbst zusammengestellt. Damit lassen sich dann alle Drucker mit Centronics-Eingang am CPC verwenden.
Aber beim ersten Probelauf gibt es eine große Überraschung. Der Drukker geht erstaunlich großzügig mit dem Papier um. Nach jeder gedruckten Zeile wird erst einmal eine Leerzeile eingeschoben.

Ursache ist folgendes:
Der CPC fügt an das Ende jeder Druckzeile die Zeichenfolge CR/LF, also die Befehlsfolge für Wagenrücklauf und Zeilenvorschub. Dadurch wird das Papier eine Zeile weiter transportiert. Zusätzlich und ohne ersichtlichen Grund ist aber noch der Pin 14 des Centronics-Anschlusses des CPC mit Masse verbunden. Das bewirkt bei den meisten Druckern einen weiteren Zeilenvorschub, so daß immer eine Leerzeile produziert wird.

Abhilfe schafft in diesem Fall die Unterbrechung der Leitung zum Pin 14 des Druckers. Nach dem Auftrennen dieser Leitung und evtl. nötigem Einstellen von Schaltern im Drucker wie z.B bei Epson sollte aber alles in Ordnung sein.

1.9.5 Der Joystickanschluß

Hauptsächlich wird der Joystickanschluß sicher so genutzt, daß er seinem Namen auch gerecht wird: als Eingang zur Abfrage eines Joysticks. Über sieben der verfügbaren 9 Anschlüsse lassen sich aber auch andere Tasten oder Schalter abfragen. Bei entsprechender Programmierung und bei Verzicht auf Interrupt und Tastatur-Abfrage könnten diese sieben Anschlüsse sogar als Ausgang betrieben werden. Die Joystickanschlüsse sind ja mit dem bidirektionalen Port des Sound Chip verbunden und könnten bei den erwähnten Einschränkungen aus als Ausgang arbeiten. Allerdings ist für Ausgabezwecke der Centronics-Port sicher einfacher zu handhaben.

Wie schon im Kapitel 1.9.1 beschrieben, werden die Joysticks als Tasten der Tastatur angesehen. Aus diesem Grund sind die benötigten sieben Eingänge des Sound Chip-Ports auf die Joystickbuchse gelegt. Zusätzlich sind noch zwei Ausgänge des BCD-Dezimal-Dekoders IC101 auf die Buchse gelegt.

Jede fünfzigstel Sekunde wird einmal komplett die Tastatur abgefragt. Dabei wird auch der Zustand der Joysticks abgefragt. Für Basic-Programme steht der Zustand der Joysticks mit der JOY(nummer)-Funktion zur Verfügung. Auch mit INKEY könnte der Zustand der Joysticks einfach be-

stimmt werden. Aber auch für Assembler-Fans gibt es die Möglichkeit, den Zustand der Joysticks einfach zu bestimmen. Die System-Routine &BB24 liefert im HL-Doppelregister den aktuellen Zustand in Bits verschlüsselt zurück. Im H-Register und im Akku erhält man beim Aufruf dieser Routine den Zustand des Joystick 0, das L-Register gilt für Joystick 1. Die Verschlüsselung der Joysticktasten erfolgt nach dem gleichen Schema wie bei der JOY (x)-Funktion, Bit 0 ist gesetzt für vorwärts, Bit 1 für rückwärts, Bit 2 für links und so weiter.

1.9.6 Der Expansion Connector

Diese Schnittstelle ist die universellste des CPC. An diesem 50-poligen Leiterplattenverbinder befinden sich neben allen Signalen des Prozessors auch noch verschiedene Steuersignale. Hier werden alle Erweiterungen des Systems angeschlossen.

Die Bedeutung der Signale 3 bis 39 ist ja bereits aus der Beschreibung des Prozessors bekannt. Darum wollen wir uns hier auf die verbleibenden Anschlüsse beschränken.

Am Pin 1 steht das Soundsignal noch einmal zur Verfügung. Allerdings ist dies Signal nur Mono, alle drei Kanäle werden direkt hier zugeführt.

Pin 2 und Pin 49 sind mit der Masse der Stromversorgung verbunden.

Eine Besonderheit ist das Signal BUS-RESET* am Pin 40. Durch Low-legen dieses Signals wird ein Reset des Systems durchgeführt.
Leider löscht der CPC beim Reset den ganzen Speicher. Darum ist dieses Signal als 'Notbremse' für abgestürzte Maschinenprogramme genau so wirkungsvoll wie das aus- und wieder einschalten.

Am Pin 41 steht das eigentliche RESET-Signal für externe Erweiterungen zur Verfügung. Beachten Sie aber, das nicht alle Bausteine mit diesem RESET-Signal versorgt werden können. Der 8255 z.B. benötigt dieses Signal invertiert.

Sehr interessant sind die beiden Signale ROMEN* und ROMDIS. Das an Pin 42 des Expansion Connectors liegende ROMEN* signalisiert bei Low-Pegel einen Zugriff auf das eingebaute 32K-Rom. Dieser Zugriff kann aber durch High-Pegel am Pin 43, ROMDIS unterbunden werden. Dadurch kann das gesamte eingebaute Rom durch externe Roms oder Eproms ersetzt werden.
Bei entsprechender Dekodierung der Adressleitungen können aber auch nur bestimmte Bereiche im eingebauten Rom ausgeblendet und ersetzt werden.

Eine ähnliche Funktion haben die beiden Signale RAMRD* und RAMDIS für Lesezugriffe auf das interne Ram. Diese an den Pins 43 und 44 liegenden Signale können genutzt werden, um z.B. bestimmte Ram-Bereiche gegen Roms oder auch Rams auszutauschen.
Die Ansteuerung externer Rams ist allerdings beim CPC nicht so ganz einfach. Hauptsächliche Schwierigkeit ist die Tatsache, daß das WR*-Signal für die internen Rams nicht vom Prozessor, sondern vom Gate Array produziert wird. Dieser Schreibimpuls kann leider durch keinen (bekannten?) Programmtrick verhindert werden, so daß ein Schreibzugriff auf ein externes Ram auch immer das interne Ram adressiert und beschreibt.

Das am Pin 46 verfügbare Signal CURSOR wird bei entsprechender Programmierung vom Video-Controller geliefert. Der CRTC verfügt ja über die Möglichkeit des Hardware-Cursors. Je nach Programmierung erscheint an diesem Ausgang ein Rechteck-Signal mit einer Frequenz von etwa 1.5 oder 3 Hertz. Aber auch ständige Low- und High-Pegel an diesem Anschluß sind programmierbar.
Nach dem Einschalten des CPC liegt hier ein ständiger Low-Pegel.

Der LPEN-Eingang (Light Pen) an Pin 47 ist direkt mit dem Light Pen-Eingang des CRTC verbunden. Dieses IC verfügt über alle nötigen Register zum Betrieb des Lightpen.
Allerdings wird die Nutzung des Light Pen besonders bei hochauflösender Grafik im CPC sehr schwierig realisierbar, da der Video-Controller zwar die MA-Adresse der momentanen Light Pen-Position liefert, aber keine Angaben über die aktuelle RA-Adresse macht. Durch den speziellen Aufbau des Video-Ram ist diese Angabe aber nötig, wenn auf dem Bildschirm mit dem Light Pen gezeichnet werden soll.

Der Eingang Pin 48 trägt die Bezeichnung EXP* und ist mit dem 8255-Port B Bit 4 verbunden. Eine externe Erweiterung kann diesen Anschluß an Masse legen und sich auf diese Weise dem Betriebssystem bemerkbar machen.

Das letzte zu erwähnende Signal am Pin 50 ist das Taktsignal des Prozessors. Dieses Signal mit einer Frequenz von 4 MHz wird z.B. als Taktsignal für den Floppy-Controller benötigt.

2 DAS BETRIEBSSYSTEM

Hinter diesem, für den Uneingeweihten nichtssagenden Namen verbirgt sich der Dreh– und Angelpunkt des ganzen Rechners. Hier ist praktisch das Stellwerk, in dem die Weichen zur Verbindung von Anwenderprogramm und Hardware gestellt werden.
Auch der Basic–Interpreter ist in diesem Zusammenhang als Programm zu sehen, welches via Betriebssystem auf die Hardware des Rechners zugreift. Dieser verbindenden Funktion wegen haben wir das Kapitel auch in die Mitte des Buches gerückt.
Der Aufbau des Betriebssystems ist logisch klar untergliedert in sog. Packs, von denen jedes einen speziellen Aufgabenbereich hat. Das beginnt auf der untersten Ebene beim *MACHINE PACK*, welches am hardwarenächsten ist und z.B. den Printer–Port, die Sound–Register usw. bedient, dann weiter über das *SCREEN PACK*, das den Bildschirm handhabt und seinerseits vom *TEXT PACK* und *GRAPHICS PACK* aufgerufen wird.
Beim genauen Hinsehen fällt auf, daß jedes Pack streng in sich geschlossen ist, und die Kommunikation mit anderen Packs über genau definierte Schnittstellen erfolgt. Darüber hinaus verfügt jedes Pack über einen eigenen Ram–Bereich als Arbeitsspeicher. Der Ansprung von Routinen erfolgt in der Regel über Vektoren im Ram, selten über die direkte Rom–Adresse.
Das legt die Vermutung nahe, daß das Betriebssystem, vermutlich der Kürze der zur Verfügung stehenden Zeit wegen, von mehreren Programmierern geschrieben wurde, jeder für ein oder mehrere Packs zuständig, nachdem man sich nur über die Schnittstellen geeinigt hat.
Wie dem auch gewesen sein mag, jedenfalls eröffnet dieser klare Aufbau und der Zugriff bis in den kleinsten Winkel über Vektoren dem Anwendungsprogrammierer ungeahnte und bisher nicht gekannte Aspekte.
Als Beispiel sei nur die Möglichkeit genannt, einen Treiber für einen echten acht Bit Drucker (wie auch immer dieser hardwaremäßig angeschlossen sein mag) zu schreiben und diesen durch Verbiegen nur des Vektors *MC WAIT PRINTER* dem gesamten System zugänglich zu machen.
Dieser Tip soll Ihnen auch als Warnung dienen: *Bedienen Sie sich ruhig der Routinen des Betriebssystems, aber nur über die Vektoren*! Es könnte ja bereits jemand anders (Rom Cartridge) einige Vektoren verstellt haben, um bestimmte Funktionen erst einmal über eigene Routinen zu leiten. Sie sähen mit Ihrem Druckertreiber ja auch ganz schön alt aus, wenn die Ausgabe für die File–# 8 direkt über $07F2 laufen würde und nicht über $BDF1.

Mit der Zeit werden Sie schon selber darauf kommen, daß eigene Programme mit minimalem Aufwand geschrieben werden können, wenn man sich nur fleißig der Vektoren bedient. Absolut neu ist, daß sogar die Arithmetikroutinen von Basic über diesen Mechanismus führen, was ei-

nerseits dazu dienen kann, eigene Berechnungen dort ausführen zu lassen, andererseits, um eigene Programme dort einzuhängen, weil Sie vielleicht eine höhere Genauigkeit wünschen, usw.

Da wir Ihnen jetzt so viel von Vektoren vorgeschwärmt haben, beginnen wir im nächsten Kapitel gleich damit.

2.1 Die Betriebssystem–Vektoren

Auf den folgenden Seiten stellen wir Ihnen die Ram–Adressen vor, über die Sie Routinen im Betriebssystem anspringen oder die Sie bei Bedarf verändern können, um bestimmte Funktionen über eigene Programme zu führen.
Die Funktion der Routine ist da mit ein paar Worten angerissen, wo man sich unter dem Namen allein nicht viel vorstellen kann. Nähere Angaben zu einigen Teilen finden Sie in der Einleitung zu den jeweiligen Packs.
Teils handelt es sich hier um komplette Routinen, die hierher kopiert wurden undin die Sie notfalls mitten hinein springen können, teils um RST1 oder RST5, gefolgt von der Inline–Address (siehe hierzu Kapitel 1.1.2), die ins Rom verweist.
Im Anhang 4.1 können Sie nachschauen, wo diese Routinen im Rom zu finden sind.

B900 KL U ROM ENABLE
B903 KL U ROM DISABLE
B906 KL L ROM ENABLE
B909 KL L ROM DISABLE
B90C KL ROM RESTORE schaltet die alte Rom–Konfiguration wieder ein.
B90F KL ROM SELECT schaltet ein Expansion–Rom (bis zu 252 sind theoretisch möglich) ein.
B912 KL CURR SELECTION Welches Expansion–Rom ist derzeit in Betrieb?
B915 KL PROBE ROM Um welche Art von Rom–Erweiterung handelt es sich?
B918 KL ROM DESELECT Vorherige Rom–Erweiterung wiederherstellen.
B91B KL LDIR
B91E KL LDDR
B921 KL POLL SYNCHRONOUS Gibt es einen Event mit höherer Priorität als die des laufenden?
B939 RST 7 INTERRUPT ENTRY CONT'D
B97C KL LOW PCHL CONT'D
B982 RST 1 LOW JUMP CONT'D
B9A8 Konfig vorber. & Sprung ausf.
B9B1 KL FAR PCHL CONT'D
B9B9 KL FAR ICALL CONT'D
B9BF RST 3 LOW FAR CALL CONT'D
BA10 KL SIDE PCHL CONT'D
BA16 RST 2 LOW SIDE CALL CONT'D
BA2E RST 5 FIRM JUMP CONT'D
BA4A KL L ROM ENABLE CONT'D
BA54 KL L ROM DISABLE CONT'D

BA5E KL U ROM ENABLE CONT'D
BA68 KL U ROM DISABLE CONT'D
BA72 KL ROM RESTORE CONT'D
BA7E KL ROM SELECT CONT'D
BA83 KL PROBE ROM CONT'D
BA8C KL ROM DESELECT CONT'D
BAA2 KL CURR SELECTION CONT'D
BAA6 KL LDIR CONT'D
BAAC KL LDDR CONT'D
BACB RST 4 RAM LAM CONT'D
BADC RAM LAM (IX) entspricht ld a,(ix)
BB00 KM INITIALIZE
BB03 KM RESET
BB06 KM WAIT CHAR Auf ein Zeichen von der Tastatur warten.
BB09 KM READ CHAR Zeichen von der Tastatur holen, falls vorhanden.
BB0C KM CHAR RETURN Zeichen im Tastaturpuffer für den nächsten Zugriff hinterlegen.
BB0F KM SET EXPAND Erweiterungsstring einrichten.
BB12 KM GET EXPAND Zeichen vom Erweiterungsstring holen.
BB15 KM EXP BUFFER Speicher für Erweiterungsstring zuweisen.
BB18 KM WAIT KEY Auf Tastendruck warten.
BB1B KM READ KEY Tastennummer holen, falls eine gedrückt wurde.
BB1E KM TEST KEY Wurde eine Taste gedrückt?
BB21 KM GET STATE Shift-Status holen.
BB24 KM GET JOYSTICK
BB27 KM SET TRANSLATE Eintrag in die Tastaturtabelle vornehmen (1. Ebene).
BB2A KM GET TRANSLATE Eintrag aus Tastaturtabelle holen (1. Ebene).
BB2D KM SET SHIFT Wie BB27, jedoch 2. Ebene.
BB30 KM GET SHIFT Wie BB2A, jedoch 2. Ebene.
BB33 KM SET CONTROL Wie BB27, jedoch 3. Ebene.
BB36 KM GET CONTROL Wie BB2A, jedoch 3. Ebene.
BB39 KM SET REPEAT Wiederholungsfunktion für bestimmte Taste setzen.
BB3C KM GET REPEAT Ist Wiederholungsfunktion für eine bestimmte Taste gesetzt?
BB3F KM SET DELAY Tastenwiederholungseinsatz und -geschwindigkeit setzen.
BB42 KM GET DELAY o.a. Parameter holen.
BB45 KM ARM BREAK Break-Taste erlauben.
BB48 KM DISARM BREAK Break-Taste verriegeln.
BB4B KM BREAK EVENT Routinen bei Drücken der Break-Taste ausführen.
BB4E TXT INITIALISE
BB51 TXT RESET

BB54 TXT VDU ENABLE Es können Zeichen auf den Bildschirm geschrieben werden.
BB57 TXT VDU DISABLE Zeichendarstellung verhindern.
BB5A TXT OUTPUT (Steuer–) Zeichen darstellen oder ausführen.
BB5D TXT WR CHAR Zeichen darstellen.
BB60 TXT RD CHAR Zeichen vom Bildschirm lesen.
BB63 TXT SET GRAPHIC Darstellung von Steuerzeichen ein– oder ausschalten.
BB66 TXT WIN ENABLE Größe des lfd. Textfensters festlegen.
BB69 TXT GET WINDOW Welche Größe hat das lfd. Textfenster?
BB6C TXT CLEAR WINDOW Lfd. Textfenster löschen.
BB6F TXT SET COLUMN
BB72 TXT SET ROW
BB75 TXT SET CURSOR
BB78 TXT GET CURSOR
BB7B TXT CUR ENABLE Cursor erlauben (Anwenderprogramm).
BB7E TXT CUR DISABLE Cursor verriegeln (Anwender).
BB81 TXT CUR ON Cursor erlauben (Betriebssystem).
BB84 TXT CUR OFF Cursor verriegeln (Betriebssytem, höhere Priorität als BB7B/BB7E).
BB87 TXT VALIDATE Cursor innerhalb des Textfensters?
BB8A TXT PLACE CURSOR Cursor ein
BB8D TXT REMOVE CURSOR Cursor aus
BB90 TXT SET PEN Vordergrundfarbe setzen.
BB93 TXT GET PEN Vordergrundfarbe?
BB96 TXT SET PAPER Hintergrundfarbe setzen.
BB99 TXT GET PAPER Hintergrundfarbe?
BB9C TXT INVERSE Lfd. Vorder– und Hintergrundfarbe vertauschen.
BB9F TXT SET BACK Transparentmodus ein / aus
BBA2 TXT GET BACK Transparentmodus?
BBA5 TXT GET MATRIX Adresse des Punktmusters eines Zeichens holen.
BBA8 TXT SET MATRIX Adresse des (anwenderdefinierten) Punktmusters eines bestimmten Zeichens setzen.
BBAB TXT SET M TABLE Startadresse und erstes Zeichen einer anwenderdefinierten Punktmatrix setzen.
BBAE TXT GET M TABLE Startadresse und erstes Zeichen einer Anwendermatrix?
BBB1 TXT GET CONTROLS Adresse der Steuerzeichen–Sprungtabelle holen.
BBB4 TXT STR SELECT Textfenster wählen.
BBB7 TXT SWAP STREAMS Die Parameter (Farben, Fenstergrenzen usw.) zweier Textfenster werden miteinander vertauscht.
BBBA GRA INITIALISE
BBBD GRA RESET
BBC0 GRA MOVE ABSOLUTE
BBC3 GRA MOVE RELATIVE

BBC6 GRA ASK CURSOR Wo ist der lfd. Grafikcursor?
BBC9 GRA SET ORIGIN
BBCC GRA GET ORIGIN
BBCF GRA WIN WIDTH Linke und rechte Begrenzung des Grafikfensters setzen.
BBD2 GRA WIN HEIGHT Obere und untere Begrenzung des Grafikfensters setzen.
BBD5 GRA GET W WIDTH Linke und rechte Begrenzung des Grafikfensters?
BBD8 GRA GET W HEIGHT Obere und untere Begrenzung des Grafikfensters?
BBDB GRA CLEAR WINDOW Grafikfenster löschen.
BBDE GRA SET PEN Schreibfarbe setzen.
BBE1 GRA GET PEN Schreibfarbe?
BBE4 GRA SET PAPER Hintergrundfarbe setzen.
BBE7 GRA GET PAPER Hintergrundfarbe?
BBEA GRA PLOT ABSOLUTE Grafikpunkt setzen (absolut)
BBED GRA PLOT RELATIVE Grafikpunkt setzen (relativ zum lfd. Cursor).
BBF0 GRA TEST ABSOLUTE Punkt gesetzt? (absolut)
BBF3 GRA TEST RELATIVE Punkt gesetzt? (relativ zum lfd. Cursor).
BBF6 GRA LINE ABSOLUTE Linie von der lfd. zur absoluten Position ziehen.
BBF9 GRA LINE RELATIVE Linie von der lfd. bis zur rel. Distanz ziehen.
BBFC GRA WR CHAR Ein Zeichen an der lfd. Grafikcursorposition schreiben.
BBFF SCR INITIALISE
BC02 SCR RESET
BC05 SCR SET OFFSET Startadresse des ersten Zeichens relativ zur Basisadresse des Videorams setzen.
BC08 SCR SET BASE Basisadresse des Videorams setzen.
BC0B SCR GET LOCATION Lfd. Bildschirmstart? (Basis + Offset).
BC0E SCR SET MODE
BC11 SCR GET MODE
BC14 SCR CLEAR Bildschirm löschen.
BC17 SCR CHAR LIMITS Größtmögliche Zeilen– und Spaltenzahl des Bildschirms holen (abhängig vom Modus).
BC1A SCR CHAR POSITION
BC1D SCR DOT POSITION
BC20 SCR NEXT BYTE Eine gegebene Bildschirmadresse um eine Zeichenposition weiterrechnen.
BC23 SCR PREV BYTE Bildschirmadresse eine Position zurückrechnen.
BC26 SCR NEXT LINE Bildschirmadresse eine Zeile weiterrechnen.
BC29 SCR PREV LINE Bildschirmadresse eine Zeile zurückrechnen.
BC2C SCR INK ENCODE

BC2F SCR INK DECODE
BC32 SCR SET INK Farbe(n) einer Ink−# zuordnen.
BC35 SCR GET INK Farbe(n) einer Ink−#?
BC38 SCR SET BORDER Rahmenfarbe(n) setzen.
BC3B SCR GET BORDER Rahmenfarbe(n)?
BC3E SCR SET FLASHING Blinkzeiten setzen.
BC41 SCR GET FLASHING Blinkzeiten?
BC44 SCR FILL BOX Vorgegebenes Fenster mit einer Farbe füllen (Positionen zeichenbezogen, mode−abhängig).
BC47 SCR FLOOD BOX Vorgegebenes Fenster mit einer Farbe füllen (Positionen sind Bildschirmadressen, mode−unabhängig).
BC4A SCR CHAR INVERT Bei einem Zeichen Vorder− und Hintergrundfarbe vertauschen.
BC4D SCR HW ROLL Bildschirm eine Zeile auf oder ab (hardwaremäßig).
BC50 SCR SW ROLL Bildschirm eine Zeile auf oder ab (softwaremäßig).
BC53 SCR UNPACK Zeichenmatrix vergrößern (für Mode 0/1).
BC56 SCR REPACK Zeichenmatrix wieder auf Originalform stauchen.
BC59 SCR ACCESS Steuerzeichen sichtbar/unsichtbar setzen.
BC5C SCR PIXELS Punkt auf dem Bildschirm setzen.
BC5F SCR HORIZONTAL Horizontale Linie ziehen.
BC62 SCR VERTICAL Vertikale Linie ziehen.
BC65 CAS INITIALISE
BC68 CAS SET SPEED Schreibgeschwindigkeit setzen.
BC6B CAS NOISY Kassettenmeldungen ein / aus.
BC6E CAS START MOTOR
BC71 CAS STOP MOTOR
BC74 CAS RESTORE MOTOR Alten Motorzustand wiederherstellen.
BC77 CAS IN OPEN
BC7A CAS IN CLOSE
BC7D CAS IN ABANDON Eingabedatei sofort schließen.
BC80 CAS IN CHAR Zeichen lesen (aus dem Puffer).
BC83 CAS IN DIRECT Gesamte Datei in den Speicher ziehen.
BC86 CAS RETURN Zuletzt gelesenes Zeichen wieder zurück in den Puffer.
BC89 CAS TEST EOF Dateiende?
BC8C CAS OUT OPEN
BC8F CAS OUT CLOSE
BC92 CAS OUT ABANDON Ausgabedatei sofort schließen.
BC95 CAS OUT CHAR Zeichen schreiben (in den Puffer).
BC98 CAS OUT DIRECT Definierten Speicherbereich auf Kassette schreiben (nicht über Puffer).
BC9B CAS CATALOG
BC9E CAS WRITE Block schreiben.
BCA1 CAS READ Block lesen.
BCA4 CAS CHECK Block auf dem Band mit Speicherinhalt vergleichen.

BCA7 SOUND RESET
BCAA SOUND QUEUE Ton an die Warteschlange anhängen.
BCAD SOUND CHECK Noch Platz in der Warteschlange?
BCB0 SOUND ARM EVENT Eventblock für den Fall 'scharf machen', daß in der Warteschlange ein Platz frei wird.
BCB3 SOUND RELEASE Töne erlauben.
BCB6 SOUND HOLD Töne sofort anhalten.
BCB9 SOUND CONTINUE Zuvor angehaltene Töne weiter bearbeiten.
BCBC SOUND AMPL ENVELOPE Lautstärkehüllkurve einrichten.
BCBF SOUND TONE ENVELOPE Tonhüllkurve einrichten.
BCC2 SOUND A ADDRESS Adresse einer Lautstärkehüllkurve hölen.
BCC5 SOUND T ADDRESS Adesse einer Tonhüllkurve holen.
BCC8 KL CHOKE OFF Kernel rücksetzen.
BCCB KL ROM WALK Irgendwelche Rom–Erweiterungen?
BCCE KL INIT BACK Rom–Erweiterung einhängen.
BCD1 KL LOG EXT Residente Erweiterung einhängen.
BCD4 KL FIND COMMAND Befehl in allen eingehängten Speicherbereichen suchen.
BCD7 KL NEW FRAME FLY Eventblock einrichten und einhängen.
BCDA KL ADD FRAME FLY Eventblock einhängen.
BCDD KL DEL FRAME FLY Eventblock aushängen.
BCE0 KL NEW FAST TICKER Wie BCD7.
BCE3 KL ADD FAST TICKER Wie BCDA.
BCE6 KL DEL FAST TICKER Wie BCDD.
BCE9 KL ADD TICKER Tickerblock einrichten und einhängen.
BCEC KL DEL TICKER Tickerblock aushängen.
BCEF KL INIT EVENT Eventblock einrichten
BCF2 KL EVENT Eventblock 'kicken'.
BCF5 KL SYNC RESET Sync Pending Queue löschen.
BCF8 KL DEL SYNCHRONOUS Bestimmten Block aus der Pending Queue löschen.
BCFB KL NEXT SYNC Der nächste bitte.
BCFE KL DO SYNC Eventroutine ausführen.
BD01 KL DONE SYNC Eventroutine fertig.
BD04 KL EVENT DISABLE
BD07 KL EVENT ENABLE
BD0A KL DISARM EVENT Eventblock verriegeln (Zähler negativ).
BD0D KL TIME PLEASE
BD10 KL TIME SET
BD13 MC BOOT PROGRAM Setzt das Betriebssystem zurück und übergibt die Steuerung einer Routine in (hl).
BD16 MC START PROGRAM
BD19 MC WAIT FLYBACK Strahlrücklauf abwarten.
BD1C MC SET MODE
BD1F MC SCREEN OFFSET
BD22 MC CLEAR INKS
BD25 MC SET INKS

BD28 MC RESET PRINTER
BD2B MC PRINT CHAR Zeichen drucken wenn möglich.
BD2E MC BUSY PRINTER Drucker noch tätig?
BD31 MC SEND PRINTER Zeichen drucken (warten, bis es geht).
BD34 MC SOUND REGISTER Sound Controller mit Daten versorgen.
BD37 JUMP RESTORE Alle Sprungvektoren initialisieren.

Die folgenden Vektoren werden von BASIC benutzt.

BD3A EDIT
BD3D FLO Variable von (de) ⇨ (hl) kopieren
BD40 FLO Int ⇨ Flo
BD43 FLO 4-Byte-Wert ⇨ Flo
BD46 FLO Flo ⇨ Int
BD49 FLO Flo ⇨ Int
BD4C FLO FIX
BD4F FLO INT
BD52 FLO
BD55 FLO Zahl mit 10↑a multiplizieren.
BD58 FLO Addition
BD5B FLO Subtraktion
BD5E FLO Subtraktion
BD61 FLO Multiplikation
BD64 FLO Division
BD67 FLO Zahl mit 2↑a multiplizieren.
BD6A FLO Vergleich
BD6D FLO Vorzeichenwechsel
BD70 FLO SGN
BD73 FLO DEG / RAD
BD76 FLO PI
BD79 FLO SQR
BD7C FLO Potenzierung
BD7F FLO LOG
BD82 FLO LOG10
BD85 FLO EXP
BD88 FLO SIN
BD8B FLO COS
BD8E FLO TAN
BD91 FLO ATN
BD94 FLO 4-Byte-Wert * 256 ⇨ Flo
BD97 FLO RND Init
BD9A FLO Set RND Seed
BD9D FLO RND
BDA0 FLO Letzten RND-Wert holen.
BDA3 INT
BDA6 INT
BDA9 INT Vorzeichen in b übernehmen.

BDAC INT Addition
BDAF INT Subtraktion
BDB2 INT Subtraktion
BDB5 INT Multiplikation mit Vorzeichen
BDB8 INT Division mit Vorzeichen
BDBB INT MOD
BDBE INT Multiplikation ohne Vorzeichen
BDC1 INT Division ohne Vorzeichen
BDC4 INT Vergleich
BDC7 INT Vorzeichenwechsel
BDCA INT SGN

Hier beginnen die sog. *INDIRECTIONS*. Das sind Sprünge ins Betriebssystem, die nicht global versorgt werden, sondern individuell von jedem Pack, wenn dessen *RESET* oder *INITALISE* durchlaufen wird.

BDCD TXT DRAW CURSOR Cursor auf den Bildschirm.
BDD0 TXT UNDRAW CURSOR Cursor aus.
BDD3 TXT WRITE CHAR Zeichen auf den Bildschirm.
BDD6 TXT UNWRITE Zeichen vom Bildschirm lesen.
BDD9 TXT OUT ACTION Zeichen abbilden oder ausführen.
BDDC GRA PLOT Punkt setzen.
BDDF GRA TEST Punkt?
BDE2 GRA LINE Linie ziehen.
BDE5 SCR READ Punkt vom Bildschirm holen.
BDE8 SCR MODE CLEAR Bildschirm mit Ink#0 löschen.
BDEE KM TEST BREAK Break–Taste gedrückt?
BDF1 MC WAIT PRINTER Zeichen zum Drucker schicken.

2.2 Das Betriebssystem RAM

Hier finden Sie eine Auflistung des Betriebssystem–Ram, soweit wir die Bedeutung der einzelnen Adressen herausgefunden haben.
Voraussetzung für die Benutzung 'on the fly' ist allerdings, daß Sie sich über die Wirkung von Manipulationen vorher genau im Klaren sind, denn wie Sie sehen spielt die Musik fast ausschließlich hier, d.h. alles, was im Betriebssystem Rang und Namen hat, mauschelt in dieser Ecke, wobei es sich nicht nur um bedeutungslose Flags oder Pointer handelt, sondern zum Teil um wesentlich massivere Dinge wie z.B. die Sprungtabelle des *TEXT SCREEN*.
Wir sehen natürlich ein, und dazu haben Sie sich ja auch dieses Buch gekauft, daß so etwas geradezu zum Probieren herausfordert. Also bitte, 'ran an den Speck. Vergessen Sie nur nicht, vorher **Ihr gerade eingetipptes Programm** in Deckung zu bringen; es **könnte Schaden nehmen**.

B08B	Basic Stackpointer
B08D	Zeiger Stringsstart
B08F	Zeiger Stringende
B09A	Zeiger Stringdescr. Stack
B09C	Stringdescr. Stack
B0BA	Stringdescriptor
B0C1	Variablentyp
B0C2	INTvar / AdrFLOvar / PointSTRdesc
B100	KL Start Int Pending Queue
B104	KL div. Flags f. Int. Rout.
B105	KL sp save
B187	KL Timer low
B189	KL Timer high
B18B	KL Timerflag
B18C	KL Start Frame Fly Chain
B18E	KL Start Fast Ticker Chain
B190	KL Start Ticker Chain
B192	KL Count for Ticker
B193	KL Start Sync Pending Queue
B195	KL Priorität lfd. Event
B196	KL auszuführender Befehl
B1A8	KL lfd. Exp.–Rom
B1A9	KL Einsprung lfd. Rom
B1AB	KL lfd. Rom–Konfig.
B1C8	SCR curr. Screen Mode
B1CA	SCR Adr. Screen Start
B1CB	SCR High Byte Screen Start
B1CC	SCR Write Indirection
B1CF	SCR Bit Masken abh. v. Mode
B1D7	SCR Flash Periods

B1D8 SCR Flash Period 1. Farbe
B1D9 SCR Farbspeicher 2. Farben
B1EA SCR Farbspeicher 1. Farben
B1FB SCR Flag lfd. Farbsatz
B1FD SCR curr. Flash Period
B1FE SCR Event Block: Set Inks
B20C TXT lfd Bildschirmfenster
B20D TXT Start Params Fenster 0
B21C TXT Params Fenster 1
B22B TXT Params Fenster 2
B23A TXT Params Fenster 3
B249 TXT Params Fenster 4
B258 TXT Params Fenster 5
B267 TXT Params Fenster 6
B276 TXT Params Fenster 7
B285 TXT lfd Cursor Pos.(Row,Col)
B287 TXT Fenst.Flag(0=ges. Bildsch.)
B288 TXT lfd Fenster oben
B289 TXT lfd Fenster links
B28A TXT lfd Fenster unten
B28B TXT lfd Fenster rechts
B28C TXT lfd Roll Count
B28D TXT lfd Cursor Flag
B28E TXT VDU Flag (0=disabled)
B28F TXT lfd Pen
B290 TXT lfd Paper
B291 TXT lfd Background Mode
B293 TXT Graph Char Write Mode (0=disabl)
B294 TXT 1. Zeichen User Matrix
B296 TXT Adr. User Matrix
B2B8 TXT Zeichenzähler Control Buffer
B2B9 TXT Start Control Buffer
B2C3 TXT Sprungtabelle Steuerzeichen
B328 GRA X Origin
B32A GRA Y Origin
B32C GRA lfd X Koord.
B32E GRA lfd Y Koord.
B330 GRA X Koord GRA Fenster links
B332 GRA X Koord GRA Fenster rechts
B334 GRA Y Koord GRA Fenster oben
B336 GRA Y Koord GRA Fenster unten
B338 GRA Pen
B339 GRA Paper
B342 GRA Rechenpuffer X Koord
B344 GRA Rechenpuffer Y Koord
B4DE KM Exp. String Pointer
B4E0 KM Put Back Buffer

B4E1	KM Adr. Start Exp Buffer
B4E3	KM Adr. Ende Exp Buffer
B4E5	KM Adr. Start freier Exp Buffer
B4E7	KM Shift Lock State
B4E8	KM Caps Lock State
B4E9	KM Delay
B4EB	KM Key State Map
B4ED	KM Key 16...23
B4F1	KM Joystick 1
B4F4	KM Joystick 0
B4F5	KM währ. Scan gedr. Keys
B4FF	KM Multihit Kontr. zu B4F5
B50D	KM Break Event Block
B541	KM Adr. Key Translation Table
B543	KM Adr. Key SHIFT Table
B545	KM Adr. Key CTRL Table
B547	KM Adr. der Repeat Tabelle
B551	SOUND alte Sound Akt. (nach HOLD)
B552	SOUND lfd Sound Aktivität
B555	SOUND Sound Event Block
B55C	SOUND Params Kanal A
B59B	SOUND Params Kanal B
B5DA	SOUND Params Kanal C
B60A	SOUND Lautstärke Hüllkurven
B6FA	SOUND Ton Hüllkurven
B800	CAS Cass. Message Flag
B802	CAS Input Buffer Status
B803	CAS Adr. Start Input Buffer
B805	CAS Pointer Input Buffer
B807	CAS File Header Input
B847	CAS Output Buffer Status
B848	CAS Adr. Start Output Buffer
B84A	CAS Pointer Output Buffer
B84C	CAS File Header Output
B8D1	CAS Cass. Speed
B8DD	EDIT Insert Flag

2.4 Nutzung von Betriebssystemroutinen

Der CPC enthält mehrere hundert z.T. sehr sinnvolle und für den Programmierer gut zu benutzende Routinen oder Funktionen. So gibt es Routinen zur Abfrage der Tastatur, zum Ausgeben eines Zeichens auf dem Bildschirm, zum Verwalten der Windows oder zum Betrieb des Druckers.
Trotz der Fülle der im Betriebssystem vorhandenen Funktionen gibt es aber Dinge, die der CPC von Haus aus nicht kann. So fehlt z.B. die Möglichkeit, den Inhalt des Bildschirms, Text oder Grafik, auf einen angeschlossenen Drucker auszugeben.

Diese '**Hardcopy**' genannte Möglichkeit wollen wir Ihnen an zwei Beispielen zeigen. Im ersten Beispiel geht es dabei um eine ausschließliche Texthardcopy, die mit jedem angeschlossenen Drucker funktioniert, die zweite Hardcopy–Routine ermöglicht den Ausdruck aller Zeichen einschließlich der CPC–Grafikzeichen. Auch in hochauflösender Grafik erstellte Bilder lassen sich mit dieser Routine ausdrucken.
Als Drucker wurde von uns der NLQ 401 gewählt. Dieser preiswerte Drucker ist, was seinen Vorrat an Steuerzeichen angeht, erstaunlich kompatibel mit den Epson–Druckern MX/RX/FX. Dadurch laufen die beiden Programme auch auf den erwähnten Epson–Druckern (und allen sonstigen Kompatiblen) ohne Anpassung.

Am Ende dieses Kapitels haben Sie nicht nur zwei schnelle Hardcopy–Routinen, sondern auch einen Einblick in die Nutzung einiger Betriebssystem–Routinen.

Um den Bildschirminhalt auf einen angeschlossenen Drucker auszugeben, muß man die Zeichen zeilenweise vom Bildschirm lesen und drukken. Durch den speziellen Aufbau des Video–Ram kann man aber leider nicht direkt die Zeichen auslesen.

Statt dessen muß über den 'Umweg' einer Betriebssystem–Routine das Zeichen an der momentanen Cursor–Position bestimmt werden. Diese Routine (**TXT RD CHAR, &BB60**) übergibt das Zeichen im Akku und setzt das Carry–Flag, wenn ein Zeichen gefunden wurde. Befindet sich an der Cursor–Position dagegen kein Zeichen des CPC–Zeichensatzes, dann enthält der Akku eine Null, das Carry–Flag ist gelöscht.

Des weiteren wird eine Routine benötigt, die uns den Cursor positioniert, damit wir die Zeichen nacheinander lesen können. Diese Funktion wird durch **TXT SET CURSOR, &BB75**, ausgeführt. Wird diese Adresse aufgerufen, so wird der Inhalt des H–Registers als Spalte, der Inhalt von L als Zeile interpretiert. Dabei ist die linke obere Schreibstelle durch &0101 adressierbar.

Allerdings gibt es an dieser Stelle ein kleines Problem. Nachdem wir den Cursor mit unserer Abfrage des Bildschirms einmal über den gesamten Bildschirm gejagt haben, sollte er hinterher wieder an seinem ursprünglichen Platz stehen. Dazu müssen wir vor dem ersten Positionieren die Stelle des Cursors ermitteln und merken.
Das kann mittels **TXT GET CURSOR, &BB78**, geschehen. Nach dem Aufruf von TXT GET CURSOR enthält das HL–Registerpaar die derzeitige Cursor–Position. Diesen Wert müssen wir uns merken und zum Abschluß der Hardcopy wieder restaurieren.

Die mittels TXT RD CHAR erhaltenen Zeichen müssen an den Drucker ausgegeben werden. Dazu dient uns **MC SEND PRINTER** mit dem Einsprung bei **&BD31**. Wird diese Adresse aufgerufen, dann wird das im Akku befindliche Zeichen auf den Printerport ausgegeben.

Allerdings erwartet MC SEND PRINTER, daß der Drucker empfangsbereit ist. Ob das der Fall ist, stellt **MC BUSY PRINTER, &BD2E**, für uns fest. Ist der Drucker nicht empfangsbereit, nicht eingeschaltet oder nicht angeschlossen, dann kommt MC BUSY PRINTER mit gesetztem Carry–Flag zurück. In diesem Fall muß diese Routine solange aufgerufen werden, bis das Carry gelöscht ist. Dann kann mit MC SEND PRINTER das gewünschte Zeichen ausgegeben werden.

Nun kann aber ja auch der Fall auftreten, daß eine einmal ausgelöste Hardcopy nicht zuende ausgedruckt werden soll. In unseren Programmen wollen wir durch Drücken der 'DEL'–Taste die Hardcopy abbrechen können. Also benötigen wir eine weitere Routine, die feststellt, ob diese Taste gedrückt ist.
Diese Aufgabe kann z.B. von **KM TEST KEY, &BB1E**, ausgeführt werden. wird diese Adresse mit einem gültigen Tastencode im Akku aufgerufen, dann ist nach dem Rücksprung das Zero–Flag gelöscht, wenn die entsprechende Taste gedrückt ist. Andernfalls ist das Zero–Flag gesetzt.

Damit haben wir eigentlich alle System–Routinen, um eine Hardcopy zu schreiben. Aber spätestens beim eifrigen Programmieren wird uns auffallen, daß wir ja garnicht wissen, ob zum Zeitpunkt der Hardcopy 20, 40 oder gar 80 Zeichen dargestellt werden.
Gut, man könnte sagen: Diese Hardcopy funktioniert nur im Bildschirmmodus x. Das wäre aber eine unschöne Einschränkung, die mit geringem Programmieraufwand zu vermeiden ist.

SCR GET MODE mit dem Einsprung bei **&BC11** teilt uns im Akku und in den beiden Flags Carry und Zero mit, in welchem Darstellungsmodus sich der CPC gerade befindet. Entsprechend können wir eine Hardcopy mit der benötigten Zeichenzahl erstellen.

Doch jetzt zum eigentlichen Programm. Leser ohne Assembler können das am Schluß dieses Kapitels abgedruckte Basicprogramm verwenden. Es enthält beide Hardcopy–Programme in Data–Zeilen.

```
A100                ORG   #A100
BB78         GETCRS EQU   #BB78
BB75         SETCRS EQU   #BB75
BB60         RDCHAR EQU   #BB60
BD2E         TSTPTR EQU   #BD2E
BD31         PRTCHR EQU   #BD31
BC11         GETMOD EQU   #BC11
BB1E         TSTKEY EQU   #BB1E

A100 CD78BB         CALL  GETCRS      ;alte Cursorpos. merken
A103 2264A1         LD    (OLDPOS),HL
A106 CD11BC         CALL  GETMOD      ;Bildschirmmodus holen
A109 17             RLA   ;           Anzahl der Zeichen/20
A10A 3263A1         LD    (MODE),A    ;und merken
A10D 210101         LD    HL,#0101    ;in die obere linke Ecke
A110 2266A1         LD    (CRSPOS),HL ;den Cursor
A113 3A63A1  LL1    LD    A,(MODE)
A116 47             LD    B,A         ;1, 2 oder 4 mal
A117 0E14    LOOP   LD    C,20        ;20 Zeichen in eine Zeile
A119 C5      LLOOP  PUSH  BC
A11A E5             PUSH  HL
A11B CD75BB         CALL  SETCRS      ;Cursor platzieren
A11E E1             POP   HL
A11F CD60BB         CALL  RDCHAR      ;und das Zeichen
A122 C1             POP   BC          ;bestimmen
A123 3802           JR    C,GOOD      ;gültiges Zeichen?
A125 3E20           LD    A,32        ;sonst Leerzeichen
A127 CD58A1  GOOD   CALL  PRTOUT      ;ausgeben
A12A E5             PUSH  HL
A12B C5             PUSH  BC
A12C 3E42           LD    A,66        ;ESC gedrückt?
A12E CD1EBB         CALL  TSTKEY
A131 C1             POP   BC
A132 E1             POP   HL
A133 201C           JR    NZ,EXIT     ;wenn ja, dann Ende
A135 24      WEITER INC   H
A136 0D             DEC   C
A137 20E0           JR    NZ,LLOOP    ;20 Zeichen gedruckt?
A139 10DC           DJNZ  LOOP        ;ganze Zeile?
A13B 3E0D           LD    A,#0D       ;CR/LF ausgeben
A13D CD58A1         CALL  PRTOUT
A140 3E0A           LD    A,#0A
A142 CD58A1         CALL  PRTOUT
```

```
A145 2A66A1          LD     HL,(CRSPOS) ;Cursorpos. für
A148 2C              INC    L           ;nächste Reihe
A149 2266A1          LD     (CRSPOS),HL ;bestimmen
A14C 7D              LD     A,L
A14D FE1A            CP     26          ;25 Zeilen gedruckt?
A14F 20C2            JR     NZ,LL1
A151 2A64A1 EXIT     LD     HL,(OLDPOS) ;wenn ja, dann alte
A154 CD75BB          CALL   SETCRS      ;Cursorpos. restaurieren
A157 C9              RET    ;           undzurück
A158 C5     PRTOUT   PUSH   BC
A159 CD2EBD P1       CALL   TSTPTR      ;Printer Busy?
A15C 38FB            JR     C,P1
A15E CD31BD          CALL   PRTCHR      ;Zeichen ausgeben
A161 C1              POP    BC
A162 C9              RET
A163 00     MODE     DEFB   0
A164 0000   OLDPOS   DEFW   0000
A166 0000   CRSPOS   DEFW   0000
```

Durch die Kommentare im Listing sollte das Programm einfach zu verstehen sein. Die einzige Besonderheit stellt die Methode zur Berechnung der Anzahl pro Zeile auszugebener Zeichen dar. Darum wollen wir noch kurz darauf eingehen.

Nach dem Aufruf von SCR GET MODE enthält der Akku je nach Modus 0, 1 oder 2. Zusätzlich haben Carry– und Zero–Flag folgende Zustände:

Mode 0 = Carry 1, Zero 0
Mode 1 = Carry 0, Zero 1
Mode 2 = Carry 0, Zero 0

Durch den Befehl SLA wird der Inhalt des Akkus um ein Bit nach links verschoben. Das entspricht einer Multiplikation mit zwei. Zusätzlich wird der Zustand des Carry–Flags in das freigewordene Bit 0 des Akku übertragen, das 'herausgefallene' Bit 7 wird in das Carry übernommen.
Im Mode 0 wird die im Akku befindliche 0 rotiert. Das hat auf den Inhalt des Akkus keinen Einfluß. Da aber das vom Aufruf von SCR GET MODE gesetzte Carry–Flag in das Bit 0 des Akkus übertragen wird, enthält der Akku nach dem Befehl SLA eine 1. Diese 1 bewirkt, daß ein mal 20 Zeichen in einer Zeile Ausgedruckt werden.
Im Mode 1 enthält der Akku eine 1, das Carry ist in diesem Mode gelöscht. Nach dem Befehl SLA enthält der Akku eine 2. Somit werden 2 mal 20 Zeichen in einer Zeile ausgegeben. Analog sind die Verhältnisse im Mode 2. Allerdings ist das Ergebnis des Befehls SLA eine 4 im Akku, welche 4 mal 20 gleich 80 Zeichen in einer Druckzeile bewirken.

Etwas anders sind die Verhältnisse, wenn es darum geht, eine Grafik–Hardcopy zu erzeugen. Hier können uns die Routinen TXT SET CURSOR und TXT RD CHAR nicht weiterhelfen.
Als erste Aktion wird mit **GRA INITIALISE** der Grafik–Modus eingeschaltet. Danach bestimmen wir mit **GRA GET PAPER** die Farbnummer des Hintergrundes. Mit diesem Wert werden alle Punkte des Bildschirms verglichen. Ist die Farbe eines Pixels vom Hintergrund verschieden, dann wird auf dem Papier ein Punkt erzeugt, erhalten wir dagegen die Hintergrundfarbnummer, dann wird kein Punkt gedruckt.

Leider verfügt der CPC nur über einen 7–Bit–Druckeranschluß. Dadurch ergeben sich gewisse Komplikationen.
Zunächst bedeutet dies, daß wir mit einem Male 7 Punkte, die untereinander angeordnet sind, auf den Drucker ausgeben können. Insgesamt hat die Grafik des CPC eine vertikale Auflösung von 200 Punkten. Das läßt sich aber nicht ohne Rest durch 7 teilen. Wir behalten vier Pixelreihen am unteren Bildrand übrig, die gesondert behandelt werden müssen. Dafür gibt es aber bei der Grafik–Hardcopy keine Probleme mit unterschiedlichen Textmodi. Egal, ob 20 oder 80 Zeichen, der erzeugte Ausdruck ist immer richtig.

Ein weiteres Problem mit dem 7–Bit–Ausgang ergibt sich bei der Befehlsübermittlung an den Drucker. Das Einschalten der Grafik mit ESC L erfordert für die vorhandenen 640 Pixel pro Zeile eine Angabe, die mit dem CPC garnicht zu übertragen ist. Um die geforderte Anzahl der Grafikpunkte auf dem Drucker zu erhalten, lautet die Steuersequenz:

PRINT #8,CHR$(27);"L";CHR$(128)CHR$(2)

Der Wert 128 stellt das Problem dar. Binär ausgedrückt ist 128 ein Wert mit gesetztem achtem Bit. Alle anderen Bits sind Null. Würden wir diesen Wert an den Drucker schicken, so käme dort nur eine Null an, da das achte Bit ja als Strobe verwendet wird und nicht an den Drucker ausgegeben werden kann.

Wir haben dies Problem in der Weise umgangen, indem wir uns auf 639 Punkte in horizontaler Richtung beschränken. Das ist zwar ein Punkt weniger, als auf dem Bildschirm vorhanden ist, dadurch reduziert sich aber der erste zu übertragende Wert auf 127. Nicht sehr elegant, aber es klappt.

Bevor wir nun zum Listing der Grafik–Hardcopy kommen, müssen wir noch auf eine Besonderheit hinweisen.
Obwohl der Bildschirm physikalisch nur 200 Rasterzeilen darstellt, wird im CPC bei allen Grafik–Routinen mit einer vertikalen Auflösung von 400 Punkten gerechnet. Dadurch ergibt sich ein deutlich besseres Verhältnis

von X– zu Y–Richtung, als wenn nur mit den tatsächlich vorhandenen 200 Zeilen gerechnet würde.
Der Effekt ist gut sichtbar, wenn Sie einmal das im Basic–Handbuch zum CPC abgedruckte Programm zum Zeichnen eines Kreises ausprobieren. Der Kreis ist tatsächlich (fast) rund. Ohne die beschriebene Korrektur würde eine auf der Seite liegende Ellypse erzeugt.
Die Korrektur muß auch in unserer Hardcopy stattfinden, allerdings in der genau entgegen gesetzten Form. Auch wir ermitteln die Grafik–Koordinaten im Raster 400x640 Punkte, auf dem Drucker werden aber nur 200 Punkte in vertikaler Richtung ausgegeben, um die Verzerrungen gering zu halten.

```
A000                   ORG   #A000
BBBA          GRINIT   EQU   #BBBA
BBE7          GETPAP   EQU   #BBE7
BBF0          TSTPOI   EQU   #BBF0
BD2B          PRINTO   EQU   #BD2B
BD2E          TSTPTR   EQU   #BD2E
BB1E          TSTKEY   EQU   #BB1E
A000 CDBABB            CALL  GRINIT      ;Grafik-Modus einschalten
A003 CDE7BB            CALL  GETPAPER    ;Hintergrundfarbe bestimmen
A006 32BDA0            LD    (PAPER),A
A009 CD6CA0            CALL  INITP       ;Drucker auf 7/72 Zoll einstellen
A00C 218F01            LD    HL,399      ;wir fangen oben und
A00F 22BEA0            LD    (Y-MERK),HL
A012 110000            LD    DE,0        ;links an zu drucken
A015 3E07              LD    A,7         ;leider nur mit 7 Nadeln
A017 32C0A0            LD    (ANZAHL),A
A01A CD7CA0 LLOOP      CALL  PRTESC      ;ESC-Seq. für Grafik
A01D 0E00   LL1        LD    C,0         ;C enthält Bitmuster für Printer
A01F 3AC0A0            LD    A,(ANZAHL)
A022 47                LD    B,A         ;B = Dotreihen-Zähler
A023 E5     BYTLP      PUSH  HL
A024 D5                PUSH  DE
A025 C5                PUSH  BC
A026 CDF0BB            CALL  TSTPOINT    ;Farbe des Pixels an der
                                         ;Position (hl/de) bestimmen
A029 C1                POP   BC
A02A D1                POP   DE
A02B 21BDA0            LD    HL,PAPER
A02E BE                CP    (HL)        ;Pixelfarbe=Hintergrundfarbe?
A02F E1                POP   HL
A030 37                SCF   ;           wenn Pixel <> Paper, dann
A031 2001              JR    NZ,DOT      ;Carry-Flag setzen, sonst
A033 A7                AND   A           ;Carry loeschen
A034 CB11   DOT        RL    C           ;Carry ins unterste Bit
```

```
A036  2B               DEC   HL          ;des C-Registers schieben,
A037  2B               DEC   HL          ;HL=HL-2, nächster Punkt,
A038  10E9             DJNZ  BYTLP       ;und das ganze 7 Mal
A03A  CDAFA0           CALL  TEST        ;Sonderbehandl. der letzten
A03D  79               LD    A,C         ;Bitmuster in Akku übertragen
A03E  CDA6A0           CALL  PRINT       ;und drucken
A041  13               INC   DE
A042  E5               PUSH  HL
A043  217F02           LD    HL,639      ;eine Zeile gedruckt?
A046  37               SCF
A047  ED52             SBC   HL,DE
A049  E1               POP   HL
A04A  3805             JR    C,NXTROW
A04C  2ABEA0           LD    HL,(Y-MERK)
A04F  18CC             JR    LL1
A051  23       NXTROW  INC   HL          ;Sonderbehandl. der
A052  7C               LD    A,H         ;letzten 4
A053  B5               OR    L
A054  C8               RET   Z
A055  2B               DEC   HL
A056  110000           LD    DE,0        ;Vorbereiten der nächsten
A059  22BEA0           LD    (Y-MERK),HL ;Druckzeile
A05C  3E07             LD    A,7
A05E  BD               CP    L           ;letzte 7-er Reihe?
A05F  20B9             JR    NZ,LLOOP
A061  7C               LD    A,H
A062  B4               OR    H
A063  20B5             JR    NZ,LLOOP
A065  3E04             LD    A,4         ;dann nur noch 4Reihen
A067  32C0A0           LD    (ANZAHL),A
A06A  18AE             JR    LLOOP
A06C  3E1B     INITP   LD    A,27        ;für NLQ/MX/RX/FX
A06E  CDA6A0           CALL  PRINT       ;ESC A 7, um den richtigen
A071  3E41             LD    A,65        ;Zeilenvorschub zu bekommen
A073  CDA6A0           CALL  PRINT
A076  3E07             LD    A,7
A078  CDA6A0           CALL  PRINT
A07B  C9               RET
A07C  E5       PRTESC  PUSH  HL          ;DEL-Taste gedrückt?
A07D  3E42             LD    A,66        ;wenn ja, dann HC abbrechen
A07F  CD1EBB           CALL  TSTKEY
A082  E1               POP   HL
A083  2802             JR    Z,NOKEY     ;DEL war nicht gedrückt
A085  E1               POP   HL          ;Stack manipulieren
A086  C9               RET   ;           ;umandenRetzukommen
A087  3E0D     NOKEY   LD    A,#0D       ;CR/LF ausgeben
A089  CDA6A0           CALL  PRINT
```

```
A08C 3E0A            LD    A,10
A08E CDA6A0          CALL  PRINT
A091 3E1B            LD    A,27          ;ESC L 127 2 = Grafik
A093 CDA6A0          CALL  PRINT         ;mit 639 Punkten
A096 3E4C            LD    A,76
A098 CDA6A0          CALL  PRINT
A09B 3E7F            LD    A,127
A09D CDA6A0          CALL  PRINT
A0A0 3E02            LD    A,2
A0A2 CDA6A0          CALL  PRINT
A0A5 C9              RET
A0A6 CD2EBD PRINT    CALL  TSTPTR        ;Drucker Busy?
A0A9 38FB            JR    C,PRINT
A0AB CD2BBD          CALL  PRINTOUT      ;Zeichen drucken
A0AE C9              RET
A0AF 3AC0A0 TEST     LD    A,(ANZAHL)    ;Behandlung der letzten
A0B2 FE07            CP    7             ;vier Dotreihen
A0B4 C8              RET   Z
A0B5 AF              XOR   A
A0B6 CB11            RL    C             ;dreimal 0 über Carry
A0B8 CB11            RL    C             ;ins C-Reg. schieben
A0BA CB11            RL    C
A0BC C9              RET
A0BD 00     PAPER    DEFB  0
A0BE 0000   Y-MERK   DEFW  0000
A0C0 00     ANZAHL   DEFB  0
```

Zum Abschluß jetzt der versprochene Basic–Lader. Damit können Sie die Programme auch einsetzen, wenn Sie keinen Monitor oder Assembler besitzen.

```
100 REM Grafik-Hardcopy für cpc 464 mit NLQ/MX/RX/FX
110 REM hardcopy wird mit 'CALL &A000' aufgerufen
120 REM text-hardcopy für den cpc 464
130 REM hardcopy wird mit 'call &a100' aufgerufen
140 FOR i = &a000 TO &a0bf
150 READ byte : POKE i,byte : s = s + byte : NEXT
160 DATA &CD,&BA,&BB,&CD,&E7,&BB,&32,&BD
165 DATA &A0,&CD,&6C,&A0,&21,&8F,&01,&22
170 DATA &BE,&A0,&11,&00,&00,&3E,&07,&32
175 DATA &C0,&A0,&CD,&7C,&A0,&0E,&00,&3A
180 DATA &C0,&A0,&47,&E5,&D5,&C5,&CD,&F0
185 DATA &BB,&C1,&D1,&21,&BD,&A0,&BE,&E1
190 DATA &37,&20,&01,&A7,&CB,&11,&2B,&2B
195 DATA &10,&E9,&CD,&AF,&A0,&79,&CD,&A6
200 DATA &A0,&13,&E5,&21,&7F,&02,&37,&ED
205 DATA &52,&E1,&38,&05,&2A,&BE,&A0,&18
```

```
210 DATA &CC,&23,&7C,&B5,&C8,&2B,&11,&00
215 DATA &00,&22,&BE,&A0,&3E,&07,&BD,&20
220 DATA &B9,&7C,&B4,&20,&B5,&3E,&04,&32
225 DATA &C0,&A0,&18,&AE,&3E,&1B,&CD,&A6
230 DATA &A0,&3E,&41,&CD,&A6,&A0,&3E,&07
235 DATA &CD,&A6,&A0,&C9,&E5,&3E,&42,&CD
240 DATA &1E,&BB,&E1,&28,&02,&E1,&C9,&3E
245 DATA &0D,&CD,&A6,&A0,&3E,&0A,&CD,&A6
250 DATA &A0,&3E,&1B,&CD,&A6,&A0,&3E,&4C
255 DATA &CD,&A6,&A0,&3E,&7F,&CD,&A6,&A0
260 DATA &3E,&02,&CD,&A6,&A0,&C9,&CD,&2E
265 DATA &BD,&38,&FB,&CD,&2B,&BD,&C9,&3A
270 DATA &C0,&A0,&FE,&07,&C8,&AF,&CB,&11
275 DATA &CB,&11,&CB,&11,&C9,&00,&00,&00
280 IF s <> 23767 THEN PRINT "error in grafik-hc" : END
290 PRINT "grafik-hc korrekt geladen"
300 FOR i = &a100 TO &a162 : s = 0
310 READ byte : POKE i,byte : s = s + byte : NEXT
320 DATA &CD,&78,&BB,&22,&64,&A1,&CD,&11
325 DATA &BC,&17,&32,&63,&A1,&21,&01,&01
330 DATA &22,&66,&A1,&3A,&63,&A1,&47,&0E
335 DATA &14,&C5,&E5,&CD,&75,&BB,&E1,&CD
340 DATA &60,&BB,&C1,&38,&02,&3E,&20,&CD
345 DATA &58,&A1,&E5,&C5,&3E,&42,&CD,&1E
350 DATA &BB,&C1,&E1,&20,&1C,&24,&0D,&20
355 DATA &E0,&10,&DC,&3E,&0D,&CD,&58,&A1
360 DATA &3E,&0A,&CD,&58,&A1,&2A,&66,&A1
365 DATA &2C,&22,&66,&A1,&7D,&FE,&1A,&20
370 DATA &C2,&2A,&64,&A1,&CD,&75,&BB,&C9
375 DATA &C5,&CD,&2E,&BD,&38,&FB,&CD,&31
380 DATA &BD,&C1,&C9
390 IF s <> 11873 THEN PRINT "error in text-hc" : END
400 PRINT "text-hc korrekt geladen"
```

2.4 Die Behandlung von Interrupts im Betriebssystem

Die wohl schnellste und leistungsfähigste Möglichkeit, innerhalb eines Betriebssystems auf bestimmte Ereignisse zu reagieren, ist die Interrupt–Technik.
Sie wissen sicher, was das ist. Falls nicht, hier das Wesentliche in dürren Worten:
Ein Interrupt (auf Deutsch: eine Unterbrechung) ist in der Regel ein Hardwareereignis, welches ein laufendes Programm über sein Auftreten informiert. Abhängig von diesem Ereignis soll die Software jetzt zugeordnete Aktionen durchführen, und zwar, je nach Dringlichkeit, möglichst schnell. Eine derartige Aktivität ist z.B. das Scrollen des Bildschirms während der Dunkelphase des Elektronenstrahls, damit es für den Betrachter möglichst flimmerfrei über die Bühne geht.
Diese Interupttechnik bietet den Vorteil, den übrigen Programmablauf wirklich nur für eine notwendige Aktion zu unterbrechen, so daß die Software nicht dauernd nachschauen muß, ob nun etwas passiert ist oder nicht.
Es gibt naturgemäß viele Möglichkeiten, eine solche Fähigkeit in ein Betriebssystem zu integrieren (wie böse Zungen behaupten, ebenso viele, wie es Programmierer gibt), aber wir müssen zugeben, daß uns eine solche Variante, wie sie uns im CPC 464 geboten wird, noch nicht untergekommen ist.

Es handelt sich hier um einen raffinierten Mix von Hardwareinterrupt (Unterbrechung, wenn nötig) und Polling (regelmäßig nachschauen, was los ist).
Wie dringend eine 'Anfrage' behandelt werden soll, entscheidet der Programmierer der zugehörigen Routine. Im Klartext:
Es gibt in der Maschine nur einen einzigen Interrupt, und das ist der Timer (im System 'Fast Ticker' genannt), der alle 1/300s eine Unterbrechung erzeugt. Alles Weitere ergibt sich hieraus, wie Sie sehen werden.

Es ist hier an der Zeit, einige neue Begriffe einzuführen, auf die Sie ab hier und auch im Rom–Listing öfter stoßen werden.
1. *EVENT* bedeutet ganz einfach *Ereignis*. Verstehen Sie in diesem Zusammenhang bitte eine Art softwaregesteuerten Interrupt.
2. *FRAME FLYBACK* ist nichts anderes als der oben erwähnte *Strahlrücklauf des Bildschirmes*, was alle 1/50s geschieht.
3. *TICKER* ist ein *Vielfaches des Fast Ticker* und erscheint ebenfalls alle 1/50s.

Das Ganze wird einfach so gehandhabt, daß der Programmierer, also u.U. Sie, bestimmt, welche Routinen seines Programmes wie oft zum Zeitpunkt Frame Flyback, Ticker oder gar Fast Ticker angesprungen werden sollen, und zwar automatisch, also ohne sein weiteres Zutun. Als Vorbe-

reitung dazu ist dem Betriebssystem lediglich, neben ein paar weiteren Kleinigkeiten, einmal die Adresse der Routine(n) mitzuteilen. Das Weitere findet sich.
Diese bereitzustellende Information nennt sich *EVENT BLOCK*. Hierin ist hinterlegt, wie oft und wann die Routine aufgerufen werden soll, ob vor evtl. weiteren Routinen (Prioritätensteuerung) oder ob es damit Zeit hat, usw.

Bei jedem Eintritt von Ticker, Fast Ticker oder Frame Fly schaut das Betriebssystem nach, ob es zugehörige Event–Blocks gibt. Falls ja, werden sie, entsprechend ihrer Priorität, aufgerufen. Einige Event–Blocks gibt es übrigens immer, z.B. die Aktion, zum Zeitpunkt Frame Fly das Farbregister zu versorgen.

Die einem bestimmten Ereignis zugeordneten Blocks sind durch Pointer miteinander verkettet, so daß sich das Betriebssystem von einem zum anderen durchhangeln kann. Demzufolge ist es ohne Bedeutung, an welcher Adresse ein solcher Block steht, solange er sich nur *in den zentralen 32k des Ram* befindet. Diese kleine Einschränkung muß deswegen gemacht werden, da dieser Bereich der einzige ist, auf den jederzeit, unabhängig von der übrigen Rom–Konfiguration, zugegriffen werden kann.
Soll ein solcher Block ausgeführt werden, so wird er in eine andere Kette eingereiht, die sog. *Pending Queue*. Dieser Vorgang heißt *Kicken*.
Die Pending Queue wird am Ende der systemeigenen Interrupt–Routine abgearbeitet. Sie sagen sich sicher, daß ein vorhandener Block selbstverständlich ausgeführt werden soll, wozu also noch extra in eine besondere Schlange einreihen?
Nun, so selbstverständlich ist das nicht, denn Sie haben durchaus die Möglichkeit, die Behandlung eines Blockes für eine Weile auszusetzen, ohne daß Sie Ihn aus der Primärqueue ausketten müssen, was übrigens bei Event–Blocks der Ticker–Queue besonders komfortabel zu machen ist.
Übrigens: Nicht daß Sie glauben, es gäbe in dem Rechner nur diesen Timer–Interrupt. Hardware–Freaks haben ohne weiteres die Möglichkeit, via Expansion–Bus einen Interrupt zu erzeugen (asynchron), nur sollte dann auch eine entsprechende Routine vorhanden sein, die den zugehörigen Event–Block 'kickt'.

Werden wir doch einmal konkret. Was ist zu tun, wenn von diesem Mechanismus Gebrauch gemacht werden soll?
Zunächst wird natürlich ein Event–Block angelegt, dessen Aufbau wie folgt vorgeschrieben ist. Allen Event–Arten ist folgender Teil gemeinsam:

Byte 0+1 Kettungsadresse für die Pending Queue. Dieses Feld darf nur vom Betriebssystem versorgt werden!

Byte 2 Zähler
Solange der Zähler > 0 ist, verbleibt der Block in der Pending

Queue, d.h. die Routine wird sooft ausgeführt, bis er =0 ist. Ist der Zähler < 0 (d.h. > 127), bleibt der Block in der betreffenden Kette (Ticker usw.). Auch ein Kicken führt in diesem Fall nicht zur Ausführung der Routine, was ansonsten ein Erhöhen des Zählers und damit auch den Ansprung bei nächster Gelegenheit zur Folge hätte.

Byte 3 Klasse
Bit0 = 1 = Bei der Sprungadresse handelt es sich um eine Near Address, d.h. sie liegt im zentralen Ram, bzw. unteren Rom.
Bit0 = 0 = Die Sprungadresse ist eine Far Address, also im oberen Rom zu suchen.
Die Bits 1–4 bestimmen die Priorität.
Bit5 muß immer =0 sein!
Bit6 = 1 = Express. Die Express–Events haben eine höhere Priorität als normale Events mit der höchsten Priorität.
Bit7 = 1 = Asynchroner Event. Diese Events haben keine Warteschlange, sondern werden beim Kicken (*KL EVENT*) sofort in die Interrupt Pending Queue eingereiht. Handelt es sich sogar um einen Express, wird er auf der Stelle ausgeführt, ansonsten erst am Ende der Interrupt–Routine.
Achtung: Die Routine für asynchrone Express–Events *muß* im zentralen Ram liegen!

Byte 4+5 Adresse der Routine

Byte 6 Rom Select, wenn Sprungadresse vom Typ Far ist, sonst unbenutzt.

Byte 7 Hier beginnt das Benutzerfeld, welches beliebig lang sein darf. Es kann zur Übergabe von Parametern an die Routine dienen. Beim Ansprung einer Event–Routine enthält hl die Adresse von Byte 5 des Event–Blocks, wenn es sich um eine Near Address handelte, ansonsten die Adresse von Byte 6.
Dieser Umstand ermöglicht es, mehrere Blocks für die gleiche Routine anzulegen, welche anhand der Parameter sehen kann, von welchem Block sie gerufen wurde.

Abhängig vom Typ des Events, also Ticker, Fast Ticker oder Frame Fly, werden noch zwei oder sechs Bytes dem gemeinsamen Teil vorangestellt. Im Falle Fast Ticker und Frame Fly sind es nur zwei Bytes für die Kettung (nicht verändern!) in der Fast Ticker List, bzw. Frame Fly List.
Die sechs Bytes für den Ticker haben folgende Bedeutung:

Byte 0+1 Kettung für Ticker List (nicht verändern!)

Byte 2+3 Tick Count bestimmt, wie oft ein Ticker erscheinen muß, bevor der Block einmal gekickt wird.

Byte 4+5 Reload Count gibt an, mit welchem Wert der Tick Count nach Ablauf wieder geladen werden soll.

Nachdem Sie also Ihren Block mit den Werten versorgt haben, soweit sie Ihnen bekannt sind (das sollten wenigstens die letzten 5 Bytes (Event

Count=0) des gemeinsamen Teils sein), brauchen Sie nur noch hl mit der Startadresse Ihres Blockes laden (im Falle Ticker gehört noch der Tick Count nach de und der Reload Count nach bc) und, je nach dem, die Routine *KL ADD TICKER*, *KL ADD FAST TICKER* oder *KL ADD FRAME FLY* anspringen, und ab geht die Post.
Zum Aushängen des Blocks aus der Liste benutzen Sie die Routinen *KL DEL TICKER* usw., wobei hl wieder die Adresse, diesmal die des zu entfernenden Blocks, enthalten muß.

Versuchen Sie es einmal und schauen Sie nach, wie das Betriebssystem es macht, denn immer wiederkehrende Prozesse werden auch dort über den Event−Mechanismus abgehandelt.

2.5 Das Betriebssystem – ROM – Listing

Allgemein zum Rom–Listing möchten wir noch anmerken, daß wir uns zwar die größte Mühe gegeben haben, es für Sie brauchbar herzurichten, aber es bleiben durchaus noch weiße Flecken auf unserer 'Landkarte', und zwar immer dort, wo es nicht um die Systemstruktur als solche geht, sondern ganz spezielle Funktionen durchgeführt werden. Dazu zählen der *CASSETTE MANAGER*, der *GRAPHICS MANAGER* und der *SOUND MANAGER*. Derartige Dinge sind naturgemäß schwer zu interpretieren, da der Gedankengang des jeweiligen Programmierers unmöglich nachzuvollziehen ist. Wir hoffen, daß Sie das nicht als allzu große Einschränkung empfinden. Der Nutzung der Routinen steht natürlich nichts im Wege.

Hinweise zum Ansprung bestimmter Programmteile mit den Übergabeparametern finden Sie im Vorwort zu jedem Pack, soweit sie zum häufigeren Gebrauch geeignet erscheinen.

Die Übergabeparameter aller vektorierter Routinen, brauchbar oder nicht, finden Sie im *Schneider Firmware Manual*, aus dessen englischer Version wir auch die Namen der Packs entnommen haben, um, falls Sie im Besitz desselben sind, durch Neuschöpfungen keine Verwirrung zu stiften.

2.5.1 KERNEL (KL)

Das Kernel, wie der Name schon vermuten läßt, hält die Fäden in der Hand.
So ist es für die Ablaufsteuerung zuständig, d.h. Interruptbehandlung und damit verbunden die Bearbeitung von Events, Bearbeitung der Restarts, Einhängen von Romerweiterungen und Umschalten der Speicherkonfiguration.
Für den Anwender eventuell brauchbar sind die Routinen im Zusammenhang mit dem Event–Mechanismus. Sehen Sie hierzu Kapitel 2.4.

```
0000  01897F     ld     bc,7fB9        U Rom dis., Mode 1, res Teiler
0003  ED49       out    (c),c
0005  C38005     jp     0580           RESET CONT'D

0008  C382B9     jp     B982 (0413)    RST 1 LOW JUMP CONT'D

000B  C37CB9     jp     B97C (040D)    KL LOW PCHL CONT'D

000E  C5         push   bc
000F  C9         ret                   jp (bc)

0010  C316BA     jp     BA16 (04A7)    RST 2 LOW SIDE CALL CONT'D

0013  C310BA     jp     BA10 (04A1)    KL SIDE PCHL CONT'D

0016  D5         push   de
0017  C9         ret                   jp (de)

0018  C3BFB9     jp     B9BF (0450)    RST 3 LOW FAR CALL CONT'D

001B  C3B1B9     jp     B9B1 (0442)    KL FAR PCHL CONT'D

001E  E9         jp     (hl)

001F  00         nop
0020  C3CBBA     jp     BACB (055C)    RST 4 RAM LAM CONT'D

0023  C3B9B9     jp     B9B9 (044A)    KL FAR ICALL CONT'D

0026  00         nop
0027  00         nop
0028  C32EBA     jp     BA2E (04BF)    RST 5 FIRM JUMP CONT'D

002B  00         nop
002C  ED49       out    (c),c
002E  D9         exx
002F  FB         ei

*************************************************** RST 6 USER
0030  F3         di                    RST0 nach High Kernel Restore
0031  D9         exx
0032  212B00     ld     hl,002B
0035  71         ld     (hl),c
0036  1808       jr     0040
0038  C339B9     jp     B939 (03CA)    RST 7 INTERRUPT ENTRY CONT'D

003B  C9         ret                   EXT INTERRUPT

003C  00         nop
003D  00         nop
003E  00         nop
003F  00         nop
```

```
*************************************** Bis hierher wird ins Ram kopiert
0040  CBD1          set   2,c            L Rom disable
0042  18E8          jr    002C

*************************************** Restore High Kernel Jumps
0044  214000        ld    hl,0040        003f
0047  2D            dec   l              bis
0048  7E            ld    a,(hl)         0000
0049  77            ld    (hl),a         ins Ram
004A  20FB          jr    nz,0047        kopieren
004C  3EC7          ld    a,C7           RST 0 nach
004E  323000        ld    (0030),a       0030
0051  219103        ld    hl,0391        Jump
0054  1100B9        ld    de,B900(0391)  Block
0057  01E901        ld    bc,01E9        kopieren
005A  EDB0          ldir

*********************************************** KL CHOKE OFF
005C  F3            di
005D  3AABB1        ld    a,(B1AB)       (lfd. Rom-Konfig.)
0060  ED5BA9B1      ld    de,(B1A9)      (Einsprung lfd. Rom)
0064  06C0          ld    b,C0           Firmware-
0066  2100B1        ld    hl,B100        Ram
0069  3600          ld    (hl),00        bis
006B  23            inc   hl             B1C0
006C  10FB          djnz  0069           löschen
006E  47            ld    b,a
006F  0EFF          ld    c,FF
0071  A9            xor   c              war ein Rom on ?
0072  C0            ret   nz             ja >
0073  4F            ld    c,a
0074  5F            ld    e,a
0075  57            ld    d,a
0076  C9            ret

0077  7C            ld    a,h
0078  B5            or    l
0079  79            ld    a,c
007A  2004          jr    nz,0080
007C  7D            ld    a,l            falls hl=0
007D  2106C0        ld    hl,C006        Default laden
0080  32A8B1        ld    (B1A8),a       (lfd. Exp.-Rom)
0083  32ABB1        ld    (B1AB),a       (lfd. Rom-Konfig.)
0086  22A9B1        ld    (B1A9),hl      (Einsprung lfd. Rom)
0089  21FFAB        ld    hl,ABFF        Params für
008C  114000        ld    de,0040        RST3 laden
008F  01FFB0        ld    bc,B0FF
0092  3100C0        ld    sp,C000
0095  DF            rst   3              FAR CALL
0096  A9B1          dw    B1A9
0098  C7            rst   0
```

```
**************************************************** KL TIME PLEASE
0099  F3          di
009A  ED5B89B1    ld     de,(B189)     (Timer high)
009E  2A87B1      ld     hl,(B187)     (Timer low)
00A1  FB          ei
00A2  C9          ret

****************************************************** KL TIME SET
00A3  F3          di
00A4  AF          xor    a
00A5  328BB1      ld     (B18B),a      (Timerflag)
00A8  ED5389B1    ld     (B189),de     (Timer high)
00AC  2287B1      ld     (B187),hl     (Timer low)
00AF  FB          ei
00B0  C9          ret

******************************************************* Scan Events
00B1  2187B1      ld     hl,B187       Timer low
00B4  34          inc    (hl)          update
00B5  23          inc    hl            Timer
00B6  28FC        jr     z,00B4
00B8  06F5        ld     b,F5
00BA  ED78        in     a,(c)         Port B
00BC  1F          rra                  VSYNC ?
00BD  3008        jr     nc,00C7       nein >
00BF  2A8CB1      ld     hl,(B18C)     (Start Frame Fly Chain)
00C2  7C          ld     a,h
00C3  B7          or     a
00C4  C45301      call   nz,0153       Kick Event
00C7  2A8EB1      ld     hl,(B18E)     (Start Fast Ticker Chain)
00CA  7C          ld     a,h
00CB  B7          or     a
00CC  C45301      call   nz,0153       Kick Event
00CF  CD611F      call   1F61          Scan Sound Queues
00D2  2192B1      ld     hl,B192       Count for Ticker
00D5  35          dec    (hl)
00D6  C0          ret    nz
00D7  3606        ld     (hl),06
00D9  CDB71B      call   1BB7          Update Key State Map
00DC  2A90B1      ld     hl,(B190)     (Start Ticker Chain)
00DF  7C          ld     a,h
00E0  B7          or     a
00E1  C8          ret    z
00E2  2104B1      ld     hl,B104       div. Flags f. Int. Rout.
00E5  CBC6        set    0,(hl)        Ticker Chain muß noch
00E7  C9          ret                  bearbeitet werden

00E8  2B          dec    hl
00E9  3600        ld     (hl),00
00EB  2B          dec    hl
00EC  3A01B1      ld     a,(B101)
00EF  B7          or     a
00F0  200C        jr     nz,00FE
00F2  2200B1      ld     (B100),hl     (Start Int Pending Queue)
00F5  2202B1      ld     (B102),hl
```

```
00F8  2104B1     ld    hl,B104       div. Flags f. Int. Rout.
00FB  CBF6       set   6,(hl)
00FD  C9         ret

00FE  ED5B02B1   ld    de,(B102)
0102  2202B1     ld    (B102),hl
0105  EB         ex    de,hl
0106  73         ld    (hl),e
0107  23         inc   hl
0108  72         ld    (hl),d
0109  C9         ret

010A  ED7305B1   ld    (B105),sp     (sp save)
010E  3187B1     ld    sp,B187       Timer low
0111  E5         push  hl
0112  D5         push  de
0113  C5         push  bc
0114  2104B1     ld    hl,B104       div. Flags f. Int. Rout.
0117  CB76       bit   6,(hl)
0119  281E       jr    z,0139
011B  CBFE       set   7,(hl)
011D  2A00B1     ld    hl,(B100)     (Start Int Pending Queue)
0120  7C         ld    a,h
0121  B7         or    a
0122  280E       jr    z,0132
0124  5E         ld    e,(hl)
0125  23         inc   hl
0126  56         ld    d,(hl)
0127  ED5300B1   ld    (B100),de     (Start Int Pending Queue)
012B  23         inc   hl
012C  CD0A02     call  020A
012F  F3         di
0130  18EB       jr    011D
0132  2104B1     ld    hl,B104       div. Flags f. Int. Rout.
0135  CB46       bit   0,(hl)        Ticker Queue pending ?
0137  2810       jr    z,0149        nein ⇨
0139  3600       ld    (hl),00
013B  37         scf
013C  08         ex    af,af'
013D  CD8901     call  0189          Ticker Chain bearbeiten
0140  B7         or    a
0141  08         ex    af,af'
0142  2104B1     ld    hl,B104       div. Flags f. Int. Rout.
0145  7E         ld    a,(hl)
0146  B7         or    a             noch etwas zu bearbeiten ?
0147  20D2       jr    nz,011B       ja ⇨
0149  3600       ld    (hl),00       alle Flags löschen
014B  C1         pop   bc
014C  D1         pop   de
014D  E1         pop   hl
014E  ED7B05B1   ld    sp,(B105)     sp rückladen
0152  C9         ret
```

```
*********************************************************** Kick Event
0153  5E           ld    e,(hl)
0154  23           inc   hl
0155  7E           ld    a,(hl)
0156  23           inc   hl
0157  B7           or    a
0158  CAE201       jp    z,01E2       KL EVENT
015B  57           ld    d,a
015C  D5           push  de
015D  CDE201       call  01E2         KL EVENT
0160  E1           pop   hl
0161  18F0         jr    0153         Kick Event

***************************************************** KL NEW FRAME FLY
0163  E5           push  hl
0164  23           inc   hl
0165  23           inc   hl
0166  CDD201       call  01D2         KL INIT EVENT
0169  E1           pop   hl

***************************************************** KL ADD FRAME FLY
016A  118CB1       ld    de,B18C      Start Frame Fly Chain
016D  C37303       jp    0373         Add Event

0170  118CB1       ld    de,B18C      Start Frame Fly Chain
0173  C38203       jp    0382         Delete Event

**************************************************** KL NEW FAST TICKER
0176  E5           push  hl
0177  23           inc   hl
0178  23           inc   hl
0179  CDD201       call  01D2         KL INIT EVENT
017C  E1           pop   hl

***************************************************** KL ADD FAST TICKER
017D  118EB1       ld    de,B18E      Start Fast Ticker Chain
0180  C37303       jp    0373         Add Event

****************************************************** Delete Fast Ticker
0183  118EB1       ld    de,B18E      Start Fast Ticker Chain
0186  C38203       jp    0382         Delete Event

************************************************ Ticker Chain bearbeiten
0189  2A90B1       ld    hl,(B190)    (Start Ticker Chain)
018C  7C           ld    a,h
018D  B7           or    a
018E  C8           ret   z
018F  5E           ld    e,(hl)
0190  23           inc   hl
0191  56           ld    d,(hl)
0192  23           inc   hl
0193  4E           ld    c,(hl)
0194  23           inc   hl
0195  46           ld    b,(hl)
0196  78           ld    a,b
```

```
0197  B1              or     c
0198  2816            jr     z,01B0
019A  0B              dec    bc
019B  78              ld     a,b
019C  B1              or     c
019D  200E            jr     nz,01AD
019F  D5              push   de
01A0  23              inc    hl
01A1  23              inc    hl
01A2  E5              push   hl
01A3  23              inc    hl
01A4  CDE201          call   01E2          KL EVENT
01A7  E1              pop    hl
01A8  46              ld     b,(hl)
01A9  2B              dec    hl
01AA  4E              ld     c,(hl)
01AB  2B              dec    hl
01AC  D1              pop    de
01AD  70              ld     (hl),b
01AE  2B              dec    hl
01AF  71              ld     (hl),c
01B0  EB              ex     de,hl
01B1  18D9            jr     018C

*********************************************** KL ADD TICKER
01B3  E5              push   hl
01B4  23              inc    hl
01B5  23              inc    hl
01B6  F3              di
01B7  73              ld     (hl),e
01B8  23              inc    hl
01B9  72              ld     (hl),d
01BA  23              inc    hl
01BB  71              ld     (hl),c
01BC  23              inc    hl
01BD  70              ld     (hl),b
01BE  E1              pop    hl
01BF  1190B1          ld     de,B190       Start Ticker Chain
01C2  C37303          jp     0373          Add Event

*********************************************** Delete Ticker
01C5  1190B1          ld     de,B190       Start Ticker Chain
01C8  CD8203          call   0382          Delete Event
01CB  D0              ret    nc
01CC  EB              ex     de,hl
01CD  23              inc    hl
01CE  5E              ld     e,(hl)
01CF  23              inc    hl
01D0  56              ld     d,(hl)
01D1  C9              ret
```

```
*********************************************** KL INIT EVENT
01D2  F3        di
01D3  23        inc   hl
01D4  23        inc   hl
01D5  3600      ld    (hl),00
01D7  23        inc   hl
01D8  70        ld    (hl),b
01D9  23        inc   hl
01DA  73        ld    (hl),e
01DB  23        inc   hl
01DC  72        ld    (hl),d
01DD  23        inc   hl
01DE  71        ld    (hl),c
01DF  23        inc   hl
01E0  FB        ei
01E1  C9        ret

************************************************** KL EVENT
01E2  23        inc   hl
01E3  23        inc   hl
01E4  F3        di
01E5  7E        ld    a,(hl)
01E6  34        inc   (hl)
01E7  FA0602    jp    m,0206      Event Cnt >127/<0
01EA  B7        or    a
01EB  2013      jr    nz,0200     Event Cnt >0 & <127
01ED  23        inc   hl
01EE  7E        ld    a,(hl)
01EF  2B        dec   hl
01F0  B7        or    a
01F1  F22F02    jp    p,022F      Sync Event einhängen
01F4  08        ex    af,af'
01F5  3012      jr    nc,0209
01F7  08        ex    af,af'
01F8  87        add   a,a
01F9  F2E800    jp    p,00E8
01FC  23        inc   hl
01FD  23        inc   hl
01FE  1823      jr    0223
0200  08        ex    af,af'
0201  3801      jr    c,0204
0203  FB        ei
0204  08        ex    af,af'
0205  C9        ret

0206  35        dec   (hl)
0207  18F7      jr    0200
0209  08        ex    af,af'
020A  FB        ei
020B  7E        ld    a,(hl)
020C  B7        or    a
020D  F8        ret   m
020E  E5        push  hl
020F  CD1C02    call  021C
0212  E1        pop   hl
```

```
0213  35              dec   (hl)
0214  C8              ret   z
0215  F20E02          jp    p,020E
0218  34              inc   (hl)
0219  C9              ret

************************************************ KL DO SYNC
021A  23              inc   hl
021B  23              inc   hl
021C  23              inc   hl
021D  7E              ld    a,(hl)
021E  23              inc   hl
021F  1F              rra
0220  D2B9B9          jp    nc,B9B9 (044A)  KL FAR ICALL CONT'D
0223  5E              ld    e,(hl)
0224  23              inc   hl
0225  56              ld    d,(hl)
0226  EB              ex    de,hl
0227  E9              jp    (hl)

********************************************** KL SYNC RESET
0228  210000          ld    hl,0000
022B  2294B1          ld    (B194),hl
022E  C9              ret

******************************************** Sync Event einhängen
022F  E5              push  hl
0230  47              ld    b,a            Priorität ⇨ b
0231  1196B1          ld    de,B196
0234  EB              ex    de,hl
0235  2B              dec   hl
0236  2B              dec   hl
0237  56              ld    d,(hl)         Adr. nächster
0238  2B              dec   hl             Event Block
0239  5E              ld    e,(hl)         ⇨ de
023A  7A              ld    a,d
023B  B7              or    a
023C  2807            jr    z,0245
023E  13              inc   de
023F  13              inc   de
0240  13              inc   de
0241  1A              ld    a,(de)         lfd Priorität > ge-
0242  B8              cp    b              fundene Priorität?
0243  30EF            jr    nc,0234        nein ⇨
0245  D1              pop   de
0246  1B              dec   de
0247  23              inc   hl
0248  7E              ld    a,(hl)
0249  12              ld    (de),a
024A  1B              dec   de
024B  72              ld    (hl),d
024C  2B              dec   hl
024D  7E              ld    a,(hl)
024E  12              ld    (de),a
024F  73              ld    (hl),e
```

```
0250 08              ex    af,af'
0251 3801            jr    c,0254
0253 FB              ei
0254 08              ex    af,af'
0255 C9              ret

**************************************************** KL NEXT SYNC
0256 F3              di
0257 2A93B1          ld    hl,(B193)       (Start Sync Pending Queue)
025A 7C              ld    a,h
025B B7              or    a
025C 2817            jr    z,0275
025E E5              push  hl
025F 5E              ld    e,(hl)
0260 23              inc   hl
0261 56              ld    d,(hl)
0262 23              inc   hl
0263 23              inc   hl
0264 3A95B1          ld    a,(B195)        (Priorität lfd. Event)
0267 BE              cp    (hl)
0268 300A            jr    nc,0274
026A F5              push  af
026B 7E              ld    a,(hl)
026C 3295B1          ld    (B195),a        (Priorität lfd. Event)
026F ED5393B1        ld    (B193),de       (Start Sync Pending Queue)
0273 F1              pop   af
0274 E1              pop   hl
0275 FB              ei
0276 C9              ret

**************************************************** KL DONE SYNC
0277 3295B1          ld    (B195),a        (Priorität lfd. Event)
027A 23              inc   hl
027B 23              inc   hl
027C 35              dec   (hl)
027D C8              ret   z
027E F3              di
027F F22F02          jp    p,022F          Sync Event einhängen
0282 34              inc   (hl)
0283 FB              ei
0284 C9              ret

********************************************** KL DEL SYNCHRONOUS
0285 CD8E02          call  028E            KL DISARM EVENT
0288 1193B1          ld    de,B193         Start Sync Pending Queue
028B C38203          jp    0382            Delete Event

************************************************* KL DISARM EVENT
028E 23              inc   hl
028F 23              inc   hl
0290 36C0            ld    (hl),C0
0292 2B              dec   hl
0293 2B              dec   hl
0294 C9              ret
```

```
************************************************ KL EVENT DISABLE
0295  2195B1      ld    hl,B195        Priorität lfd. Event
0298  CBEE        set   5,(hl)
029A  C9          ret

************************************************ KL EVENT ENABLE
029B  2195B1      ld    hl,B195        Priorität lfd. Event
029E  CBAE        res   5,(hl)
02A0  C9          ret

*************************************************** KL LOG EXT
02A1  E5          push  hl
02A2  ED5BA6B1    ld    de,(B1A6)
02A6  22A6B1      ld    (B1A6),hl
02A9  73          ld    (hl),e
02AA  23          inc   hl
02AB  72          ld    (hl),d
02AC  23          inc   hl
02AD  71          ld    (hl),c
02AE  23          inc   hl
02AF  70          ld    (hl),b
02B0  E1          pop   hl
02B1  C9          ret

************************************************ KL FIND COMMAND
02B2  1196B1      ld    de,B196        auszuführender Befehl
02B5  011000      ld    bc,0010
02B8  CDA6BA      call  BAA6 (0537)    Rom off & Konfig save
02BB  EB          ex    de,hl
02BC  2B          dec   hl
02BD  CBFE        set   7,(hl)
02BF  2AA6B1      ld    hl,(B1A6)
02C2  7D          ld    a,l
02C3  1810        jr    02D5
02C5  E5          push  hl
02C6  23          inc   hl
02C7  23          inc   hl
02C8  4E          ld    c,(hl)
02C9  23          inc   hl
02CA  46          ld    b,(hl)
02CB  CDF402      call  02F4
02CE  D1          pop   de
02CF  D8          ret   c
02D0  EB          ex    de,hl
02D1  7E          ld    a,(hl)
02D2  23          inc   hl
02D3  66          ld    h,(hl)
02D4  6F          ld    l,a
02D5  B4          or    h
02D6  20ED        jr    nz,02C5
02D8  0EFF        ld    c,FF
02DA  0C          inc   c
02DB  CD83BA      call  BA83 (0514)    KL PROBE ROM CONT'D
02DE  F5          push  af
02DF  E603        and   03
```

```
02E1  47        ld    b,a
02E2  CCF402    call  z,02F4
02E5  3809      jr    c,02F0
02E7  F1        pop   af
02E8  87        add   a,a
02E9  30EF      jr    nc,02DA
02EB  79        ld    a,c
02EC  B7        or    a
02ED  28EB      jr    z,02DA
02EF  C9        ret

02F0  F1        pop   af
02F1  C30B06    jp    060B          MC START PROGRAM

02F4  2104C0    ld    hl,C004
02F7  78        ld    a,b
02F8  B7        or    a
02F9  2804      jr    z,02FF
02FB  60        ld    h,b
02FC  69        ld    l,c
02FD  0EFF      ld    c,FF
02FF  CD7EBA    call  BA7E (050F)   KL ROM SELECT CONT'D
0302  C5        push  bc
0303  5E        ld    e,(hl)
0304  23        inc   hl
0305  56        ld    d,(hl)
0306  23        inc   hl
0307  EB        ex    de,hl
0308  1817      jr    0321
030A  0196B1    ld    bc,B196       auszuführender Befehl
030D  0A        ld    a,(bc)
030E  BE        cp    (hl)
030F  2008      jr    nz,0319
0311  23        inc   hl
0312  03        inc   bc
0313  87        add   a,a
0314  30F7      jr    nc,030D
0316  EB        ex    de,hl
0317  180C      jr    0325
0319  7E        ld    a,(hl)
031A  23        inc   hl
031B  87        add   a,a
031C  30FB      jr    nc,0319
031E  13        inc   de
031F  13        inc   de
0320  13        inc   de
0321  7E        ld    a,(hl)
0322  B7        or    a
0323  20E5      jr    nz,030A
0325  C1        pop   bc
0326  C38CBA    jp    BA8C (051D)   KL ROM DESELECT CONT'D
```

```
****************************************************** KL ROM WALK
0329  0E07       ld    c,07
032B  CD3203     call  0332          KL INIT BACK
032E  0D         dec   c
032F  20FA       jr    nz,032B
0331  C9         ret

******************************************************* KL INIT BACK
0332  79         ld    a,c
0333  FE08       cp    08
0335  D0         ret   nc
0336  CD7EBA     call  BA7E (050F)   KL ROM SELECT CONT'D
0339  3A00C0     ld    a,(C000)
033C  E603       and   03
033E  3D         dec   a
033F  201F       jr    nz,0360
0341  C5         push  bc
0342  CD06C0     call  C006
0345  D5         push  de
0346  23         inc   hl
0347  EB         ex    de,hl
0348  21AAB1     ld    hl,B1AA
034B  ED4BA8B1   ld    bc,(B1A8)     (lfd. Exp.-Rom)
034F  0600       ld    b,00
0351  09         add   hl,bc
0352  09         add   hl,bc
0353  73         ld    (hl),e
0354  23         inc   hl
0355  72         ld    (hl),d
0356  21FCFF     ld    hl,FFFC
0359  19         add   hl,de
035A  CDA102     call  02A1          KL LOG EXT
035D  2B         dec   hl
035E  D1         pop   de
035F  C1         pop   bc
0360  C38CBA     jp    BA8C (051D)   KL ROM DESELECT CONT'D

0363  7E         ld    a,(hl)
0364  BB         cp    e
0365  23         inc   hl
0366  7E         ld    a,(hl)
0367  2B         dec   hl
0368  2003       jr    nz,036D
036A  BA         cp    d
036B  37         scf
036C  C8         ret   z
036D  B7         or    a
036E  C8         ret   z
036F  6E         ld    l,(hl)
0370  67         ld    h,a
0371  18F0       jr    0363
```

```
*********************************************************** Add Event
0373  EB          ex    de,hl
0374  F3          di
0375  CD6303      call  0363
0378  3806        jr    c,0380
037A  73          ld    (hl),e
037B  23          inc   hl
037C  72          ld    (hl),d
037D  13          inc   de
037E  AF          xor   a
037F  12          ld    (de),a
0380  FB          ei
0381  C9          ret

******************************************************** Delete Event
0382  EB          ex    de,hl
0383  F3          di
0384  CD6303      call  0363
0387  3006        jr    nc,038F
0389  1A          ld    a,(de)
038A  77          ld    (hl),a
038B  13          inc   de
038C  23          inc   hl
038D  1A          ld    a,(de)
038E  77          ld    (hl),a
038F  FB          ei
0390  C9          ret

0391  C35EBA      jp    BA5E (04EF)   KL U ROM ENABLE CONT'D

0394  C368BA      jp    BA68 (04F9)   KL U ROM DISABLE CONT'D

0397  C34ABA      jp    BA4A (04DB)   KL L ROM ENABLE CONT'D

039A  C354BA      jp    BA54 (04E5)   KL L ROM DISABLE CONT'D

039D  C372BA      jp    BA72 (0503)   KL ROM RESTORE CONT'D

03A0  C37EBA      jp    BA7E (050F)   KL ROM SELECT CONT'D

03A3  C3A2BA      jp    BAA2 (0533)   KL CURR SELECTION CONT'D

03A6  C383BA      jp    BA83 (0514)   KL PROBE ROM CONT'D

03A9  C38CBA      jp    BA8C (051D)   KL ROM DESELECT CONT'D

03AC  C3A6BA      jp    BAA6 (0537)   KL LDIR CONT'D

03AF  C3ACBA      jp    BAAC (053D)   KL LDDR CONT'D

*********************************************** KL POLL SYCHRONOUS
03B2  3A94B1      ld    a,(B194)
03B5  B7          or    a
03B6  C8          ret   z
03B7  E5          push  hl
```

```
03B8  F3           di
03B9  2A93B1       ld     hl,(B193)          (Start Sync Pending Queue)
03BC  7C           ld     a,h
03BD  B7           or     a
03BE  2807         jr     z,03C7
03C0  23           inc    hl
03C1  23           inc    hl
03C2  23           inc    hl
03C3  3A95B1       ld     a,(B195)           (Priorität lfd. Event)
03C6  BE           cp     (hl)
03C7  E1           pop    hl
03C8  FB           ei
03C9  C9           ret

************************************** RST 7 INTERRUPT ENTRY CONT'D
03CA  F3           di
03CB  08           ex     af,af'
03CC  3833         jr     c,0401             EXT INTERRUPT ENTRY
03CE  D9           exx
03CF  79           ld     a,c
03D0  37           scf
03D1  FB           ei
03D2  08           ex     af,af'
03D3  F3           di
03D4  F5           push   af
03D5  CB91         res    2,c
03D7  ED49         out    (c),c              L Rom enable
03D9  CDB100       call   00B1               Scan Events
03DC  B7           or     a
03DD  08           ex     af,af'
03DE  4F           ld     c,a
03DF  067F         ld     b,7F
03E1  3A04B1       ld     a,(B104)           (div. Flags f. Int. Rout.)
03E4  B7           or     a
03E5  2814         jr     z,03FB
03E7  FA6AB9       jp     m,B96A (03FB)
03EA  79           ld     a,c
03EB  E60C         and    0C
03ED  F5           push   af
03EE  CB91         res    2,c
03F0  D9           exx
03F1  CD0A01       call   010A
03F4  D9           exx
03F5  E1           pop    hl
03F6  79           ld     a,c
03F7  E6F3         and    F3
03F9  B4           or     h
03FA  4F           ld     c,a
03FB  ED49         out    (c),c              alte Konfig. setzen
03FD  D9           exx
03FE  F1           pop    af
03FF  FB           ei
0400  C9           ret
```

```
*********************************************** EXT INTERRUPT ENTRY
0401  08              ex     af,af'
0402  E1              pop    hl
0403  F5              push   af
0404  CBD1            set    2,c
0406  ED49            out    (c),c          L Rom disable
0408  CD3B00          call   003B
040B  18CF            jr     03DC

*********************************************** KL LOW PCHL CONT'D
040D  F3              di
040E  E5              push   hl
040F  D9              exx
0410  D1              pop    de
0411  1806            jr     0419

*********************************************** RST 1 LOW JUMP CONT'D
0413  F3              di
0414  D9              exx
0415  E1              pop    hl
0416  5E              ld     e,(hl)
0417  23              inc    hl
0418  56              ld     d,(hl)
0419  08              ex     af,af'
041A  7A              ld     a,d
041B  CBBA            res    7,d
041D  CBB2            res    6,d
041F  07              rlca
0420  07              rlca
0421  07              rlca
0422  07              rlca
0423  A9              xor    c
0424  E60C            and    0C
0426  A9              xor    c
0427  C5              push   bc
0428  CDA8B9          call   B9A8 (0439)    Konfig vorber. & Sprung ausf.
042B  F3              di
042C  D9              exx
042D  08              ex     af,af'
042E  79              ld     a,c
042F  C1              pop    bc
0430  E603            and    03
0432  CB89            res    1,c
0434  CB81            res    0,c
0436  B1              or     c
0437  1801            jr     043A
0439  D5              push   de             Sprungadr. auf Stack
043A  4F              ld     c,a
043B  ED49            out    (c),c          Rom Konfig setzen
043D  B7              or     a
043E  08              ex     af,af'
043F  D9              exx
0440  FB              ei
0441  C9              ret                   Sprung ausführen
```

```
********************************************** KL FAR PCHL CONT'D
0442  F3          di
0443  08          ex    af,af'
0444  79          ld    a,c
0445  E5          push  hl
0446  D9          exx
0447  D1          pop   de
0448  1815        jr    045F

********************************************** KL FAR ICALL CONT'D
044A  F3          di
044B  E5          push  hl
044C  D9          exx
044D  E1          pop   hl
044E  1809        jr    0459

****************************************** RST 3 LOW FAR CALL CONT'D
0450  F3          di
0451  D9          exx
0452  E1          pop   hl
0453  5E          ld    e,(hl)
0454  23          inc   hl
0455  56          ld    d,(hl)
0456  23          inc   hl
0457  E5          push  hl
0458  EB          ex    de,hl
0459  5E          ld    e,(hl)
045A  23          inc   hl
045B  56          ld    d,(hl)
045C  23          inc   hl
045D  08          ex    af,af'
045E  7E          ld    a,(hl)
045F  FEFC        cp    FC          Rom# > 252 ?
0461  30BE        jr    nc,0421     ja ⇨
0463  06DF        ld    b,DF        Expansion Rom
0465  ED79        out   (c),a       einschalten
0467  21A8B1      ld    hl,B1A8     lfd. Exp.-Rom
046A  46          ld    b,(hl)
046B  77          ld    (hl),a
046C  C5          push  bc
046D  FDE5        push  iy
046F  3D          dec   a
0470  FE07        cp    07
0472  300F        jr    nc,0483
0474  87          add   a,a
0475  C6AC        add   a,AC
0477  6F          ld    l,a
0478  CEB1        adc   a,B1
047A  95          sub   l
047B  67          ld    h,a
047C  7E          ld    a,(hl)
047D  23          inc   hl
047E  66          ld    h,(hl)
047F  6F          ld    l,a
0480  E5          push  hl
```

```
0481  FDE1          pop   iy
0483  067F          ld    b,7F
0485  79            ld    a,c
0486  CBD7          set   2,a           L Rom disable
0488  CB9F          res   3,a           U Rom enable
048A  CDA8B9        call  B9A8 (0439)   Konfig vorber. & Sprung ausf.
048D  FDE1          pop   iy
048F  F3            di
0490  D9            exx
0491  08            ex    af,af'
0492  59            ld    e,c
0493  C1            pop   bc            alte
0494  78            ld    a,b           Expansion Rom-
0495  06DF          ld    b,DF          Konfiguration
0497  ED79          out   (c),a         wiederherstellen
0499  32A8B1        ld    (B1A8),a      (lfd. Exp.-Rom)
049C  067F          ld    b,7F
049E  7B            ld    a,e
049F  188F          jr    0430

********************************************** KL SIDE PCHL CONT'D
04A1  F3            di
04A2  E5            push  hl
04A3  D9            exx
04A4  D1            pop   de
04A5  1808          jr    04AF

****************************************** RST 2 LOW SIDE CALL CONT'D
04A7  F3            di
04A8  D9            exx
04A9  E1            pop   hl
04AA  5E            ld    e,(hl)
04AB  23            inc   hl
04AC  56            ld    d,(hl)
04AD  23            inc   hl
04AE  E5            push  hl
04AF  08            ex    af,af'
04B0  7A            ld    a,d
04B1  CBFA          set   7,d
04B3  CBF2          set   6,d
04B5  E6C0          and   C0
04B7  07            rlca
04B8  07            rlca
04B9  21ABB1        ld    hl,B1AB       lfd. Rom-Konfig.
04BC  86            add   a,(hl)
04BD  18A4          jr    0463

******************************************** RST 5 FIRM JUMP CONT'D
04BF  F3            di
04C0  D9            exx
04C1  E1            pop   hl
04C2  5E            ld    e,(hl)
04C3  23            inc   hl
04C4  56            ld    d,(hl)
04C5  CB91          res   2,c
```

```
04C7  ED49          out   (c),c          L Rom enable
04C9  ED533FBA      ld    (BA3F),de      Sprungadr. laden
04CD  D9            exx
04CE  FB            ei
04CF  CD3EBA        call  BA3E (04CF)    und ausführen
04D2  F3            di
04D3  D9            exx
04D4  CBD1          set   2,c            L Rom disable
04D6  ED49          out   (c),c
04D8  D9            exx
04D9  FB            ei
04DA  C9            ret

********************************************** KL L ROM ENABLE
04DB  F3            di
04DC  D9            exx
04DD  79            ld    a,c
04DE  CB91          res   2,c
04E0  ED49          out   (c),c          L Rom enable
04E2  D9            exx
04E3  FB            ei
04E4  C9            ret

********************************************** KL L ROM DISABLE
04E5  F3            di
04E6  D9            exx
04E7  79            ld    a,c
04E8  CBD1          set   2,c
04EA  ED49          out   (c),c          L Rom disable
04EC  D9            exx
04ED  FB            ei
04EE  C9            ret

********************************************** KL U ROM ENABLE
04EF  F3            di
04F0  D9            exx
04F1  79            ld    a,c
04F2  CB99          res   3,c
04F4  ED49          out   (c),c          U Rom enable
04F6  D9            exx
04F7  FB            ei
04F8  C9            ret

********************************************** KL U ROM DISABLE
04F9  F3            di
04FA  D9            exx
04FB  79            ld    a,c
04FC  CBD9          set   3,c
04FE  ED49          out   (c),c          U Rom disable
0500  D9            exx
0501  FB            ei
0502  C9            ret
```

```
****************************************************** KL ROM RESTORE
0503  F3          di
0504  D9          exx                        a enthält
0505  A9          xor     c                  die alte
0506  E60C        and     0C                 Konfiguration
0508  A9          xor     c
0509  4F          ld      c,a
050A  ED49        out     (c),c
050C  D9          exx
050D  FB          ei
050E  C9          ret

******************************************************* KL ROM SELECT
050F  CD5EBA      call    BA5E (04EF)        KL U ROM ENABLE CONT'D
0512  180F        jr      0523

******************************************************* KL PROBE ROM
0514  CD7EBA      call    BA7E (050F)        KL ROM SELECT CONT'D
0517  3A00C0      ld      a,(C000)
051A  2A01C0      ld      hl,(C001

****************************************************** KL ROM DESELECT
051D  F5          push    af
051E  78          ld      a,b
051F  CD72BA      call    BA72 (0503)        KL ROM RESTORE CONT'D
0522  F1          pop     af
0523  E5          push    hl
0524  F3          di
0525  06DF        ld      b,DF               Expansion Rom (# in c)
0527  ED49        out     (c),c              einschalten
0529  21A8B1      ld      hl,B1A8            lfd. Exp.-Rom
052C  46          ld      b,(hl)
052D  71          ld      (hl),c
052E  48          ld      c,b
052F  47          ld      b,a
0530  FB          ei
0531  E1          pop     hl
0532  C9          ret

***************************************************** KL CURR SELECTION
0533  3AA8B1      ld      a,(B1A8)           (lfd. Exp.-Rom)
0536  C9          ret

************************************************************* KL LDIR
0537  CDB2BA      call    BAB2 (0543)
053A  EDB0        ldir
053C  C9          ret

************************************************************* KL LDDR
053D  CDB2BA      call    BAB2 (0543)
0540  EDB8        lddr
0542  C9          ret
```

```
************************************************ Rom off & Konfig. save
0543  F3          di
0544  D9          exx
0545  E1          pop    hl             RET-Adr. manipulieren
0546  C5          push   bc             alte Konfig. merken
0547  CBD1        set    2,c            Roms
0549  CBD9        set    3,c            disable
054B  ED49        out    (c),c
054D  CDC7BA      call                  call (hl)
0550  F3          di
0551  D9          exx                   alte
0552  C1          pop    bc             Konfiguration
0553  ED49        out    (c),c          wiederherstellen
0555  D9          exx
0556  FB          ei
0557  C9          ret

0558  E5          push   hl             RET-Adr. manipulieren
0559  D9          exx
055A  FB          ei
055B  C9          ret

********************************************************* RAM LAM
055C  F3          di
055D  D9          exx
055E  59          ld     e,c
055F  CBD3        set    2,e            Roms
0561  CBDB        set    3,e            disable
0563  ED59        out    (c),e
0565  D9          exx
0566  7E          ld     a,(hl)         Byte holen
0567  D9          exx
0568  ED49        out    (c),c          alte Konfig. setzen
056A  D9          exx
056B  FB          ei
056C  C9          ret

****************************************************** RAM LAM (IX)
056D  D9          exx
056E  79          ld     a,c
056F  F60C        or     0C             Roms
0571  ED79        out    (c),a          disable
0573  DD7E00      ld     a,(ix+00)      Byte holen
0576  ED49        out    (c),c          alte Konfig. setzen
0578  D9          exx
0579  C9          ret

057A  C7          rst    0
057B  C7          rst    0
057C  C7          rst    0
057D  C7          rst    0
057E  C7          rst    0
057F  C7          rst    0
```

2.5.2 MACHINE PACK (MC)

Wie der Name schon vermuten läßt, ist dies der maschinennahe Teil des Betriebssystems.
Hier werden die diversen Schnittstellen und Peripheriebausteine wie PIO und PSG bedient. Dieses Verfahren hat den Vorteil, daß bei einer evtl. Änderung der Hardware nur das MACHINE PACK angepaßt werden muß, vergleichbar dem BIOS im CP/M.
Wegen der Hardwarenähe sind auch nur wenige Routinen für den häufigeren Gebrauch geeignet. Die folgenden haben wir herausgesucht:

MC PRINT CHAR gibt das Zeichen in **a** auf den Centronics–Port aus. Nach Rückkehr aus der Routine ist **carry** gesetzt, wenn das Zeichen erfolgreich abgesetzt wurde.

MC SOUND REGISTER ist für die Musikfans interessant. Ohne daß Sie sich mit der relativ komplizierten Datenübergabe an den PSG plagen müssen, brauchen Sie nur in **a** die gewünschte Registernummer und in **c** das Datenbyte zu übergeben.

```
**************************************************** RESET CONT'D
0580  F3        di
0581  0182F7    ld    bc,F782     Control
0584  ED49      out   (c),c
0586  0100F4    ld    bc,F400     Port A
0589  ED49      out   (c),c
058B  0100F6    ld    bc,F600     Port C
058E  ED49      out   (c),c
0590  017FEF    ld    bc,EF7F     Centronics
0593  ED49      out   (c),c
0595  06F5      ld    b,F5        Port B
0597  ED78      in    a,(c)
0599  E610      and   10          LK4 isolieren
059B  21C405    ld    hl,05C4     Ende Tabelle 60Hz
059E  2003      jr    nz,05A3     50Hz ? nein ⇨
05A0  21D405    ld    hl,05D4     Ende Tabelle 50Hz
05A3  010FBC    ld    bc,BC0F
05A6  ED49      out   (c),c       Video Reg.-Adr. laden
05A8  2B        dec   hl
05A9  7E        ld    a,(hl)
05AA  04        inc   b
05AB  ED79      out   (c),a       Video Reg. laden
05AD  05        dec   b
05AE  0D        dec   c
05AF  F2A605    jp    p,05A6
05B2  1820      jr    05D4

****************************************************** Tabelle 60Hz
05B4  3F 28 2E 8E 26 00 19 1E
05BC  00 07 00 00 30 00 C0 00

****************************************************** Tabelle 50Hz
05C4  3F 28 2E 8E 1F 06 19 1B
05CC  00 07 00 00 30 00 C0 00

05D4  115C06    ld    de,065C     Kaltstart
05D7  210000    ld    hl,0000     ist Fortsetzungs-
05DA  1832      jr    060E        Adresse

************************************************ MC BOOT PROGRAM
05DC  3100C0    ld    sp,C000
05DF  E5        push  hl
05E0  CD681E    call  1E68        SOUND RESET
05E3  F3        di
05E4  01FFF8    ld    bc,F8FF     Peripherie
05E7  ED49      out   (c),c       rücksetzen
05E9  CD5C00    call  005C        KL CHOKE OFF
05EC  E1        pop   hl
05ED  D5        push  de
05EE  C5        push  bc
05EF  E5        push  hl
05F0  CD1E1A    call  1A1E        KM RESET
05F3  CD8810    call  1088        TXT RESET
05F6  CDB10A    call  0AB1        SCR RESET
05F9  CD5EBA    call  BA5E (04EF) KL U ROM ENABLE CONT'D
```

```
05FC  E1        pop   hl
05FD  CD7507    call  0775       jp (hl)
0600  C1        pop   bc
0601  D1        pop   de
0602  3807      jr    c,060B     MC START PROGRAM
0604  EB        ex    de,hl
0605  48        ld    c,b
0606  11E806    ld    de,06E8    Ladefehler
0609  1803      jr    060E

******************************************* MC START PROGRAM
060B  112607    ld    de,0726    trifft nach 0654 auf RET
060E  F3        di
060F  ED56      im    1
0611  D9        exx
0612  0100DF    ld    bc,DF00    Palette Pointer reset
0615  ED49      out   (c),c
0617  01FFF8    ld    bc,F8FF    evtl. angeschlossene
061A  ED49      out   (c),c      Peripherie reset
061C  2100B1    ld    hl,B100    Start Int Pending Queue
061F  1101B1    ld    de,B101    Ram
0622  01FF07    ld    bc,07FF    löschen
0625  3600      ld    (hl),00
0627  EDB0      ldir
0629  01897F    ld    bc,7F89    U Rom off & L Rom on
062C  ED49      out   (c),c      Screen Mode 1
062E  D9        exx
062F  AF        xor   a
0630  08        ex    af,af'
0631  3100C0    ld    sp,C000
0634  E5        push  hl
0635  C5        push  bc
0636  D5        push  de
0637  CD4400    call  0044       Restore High Kernel Jumps
063A  CD8808    call  0888       JUMP RESTORE
063D  CDE019    call  19E0       KM INITIALISE
0640  CD681E    call  1E68       SOUND RESET
0643  CDA00A    call  0AA0       SCR INITIALISE
0646  CD7810    call  1078       TXT INITIALISE
0649  CDB015    call  15B0       GRA INITIALISE
064C  CD7023    call  2370       CAS INITIALISE
064F  CDE607    call  07E6       MC RESET PRINTER
0652  FB        ei
0653  E1        pop   hl
0654  CD7507    call  0775       jp (hl)
0657  C1        pop   bc
0658  E1        pop   hl
0659  C37700    jp    0077       U Rom initialisieren
```

```
*********************************************************** Kaltstart
065C  CD1207       call  0712       Firmennamen ausgeben
065F  CDEB06       call  06EB       Meldungen ausgeben
0662  216D06       ld    hl,066D    Einschaltmeldung
0665  CDEB06       call  06EB       Meldungen ausgeben
0668  219306       ld    hl,0693    Copyright-Meldung
066B  187E         jr    06EB       Meldungen ausgeben

****************************************************** Einschaltmeldung
066D  20 36 34 4B 20 4D 69 63       64K Mic
0675  72 6F 63 6F 6D 70 75 74      rocomput
067D  65 72 20 20 28 76 31 29      er  (v1)
0685  0D 0A 0D 0A 00 43 6F 70      .....Cop
068D  79 72 69 67 68 74 20 A4      yright ©
0695  31 39 38 34 20 41 6D 73      1984 Ams
069D  74 72 61 64 20 43 6F 6E      trad Con
06A5  73 75 6D 65 72 20 45 6C      sumer El
06AD  65 63 74 72 6F 6E 69 63      ectronic
06B5  73 20 70 6C 63 0D 0A 20      s plc..
06BD  20 20 20 20 20 20 20 20
06C5  20 20 61 6E 64 20 4C 6F        and Lo
06CD  63 6F 6D 6F 74 69 76 65      comotive
06D5  20 53 6F 66 74 77 61 72       Softwar
06DD  65 20 4C 74 64 2E 0D 0A      e Ltd...
06E5  0D 0A 00                     ...

06E8  21F406       ld    hl,06F4    Ladefehler-Meldung

**************************************************** Meldungen ausgeben
06EB  7E           ld    a,(hl)
06EC  23           inc   hl
06ED  B7           or    a
06EE  C8           ret   z
06EF  CD0014       call  1400       TXT OUTPUT
06F2  18F7         jr    06EB       Meldungen ausgeben

**************************************************** Ladefehler-Meldung
06F4  2A 2A 2A 20 50 52 4F 47      *** PROG
06FC  52 41 4D 20 4C 4F 41 44      RAM LOAD
0704  20 46 41 49 4C 45 44 20       FAILED
070C  2A 2A 2A 0D 0A 00            ***...

0712  06F5         ld    b,F5
0714  ED78         in    a,(c)      Port B
0716  2F           cpl
0717  E60E         and   0E         LK1...3 isolieren
0719  0F           rrca             /2
071A  212707       ld    hl,0727    Firmennamen
071D  3C           inc   a
071E  47           ld    b,a
071F  7E           ld    a,(hl)
0720  23           inc   hl
0721  B7           or    a
0722  20FB         jr    nz,071F
0724  10F9         djnz  071F
```

```
0726  C9                      ret

*************************************************** Firmennamen
0727  41 72 6E 6F 6C 64 00 0A              Arnold..
072F  20 41 6D 73 74 72 61 64               Amstrad
0737  00 0A 20 4F 72 69 6F 6E              .. Orion
073F  00 0A 20 53 63 68 6E 65              .. Schne
0747  69 64 65 72 00 0A 20 41              ider.. A
074F  77 61 00 0A 20 53 6F 6C              wa.. Sol
0757  61 76 6F 78 00 0A 20 53              avox.. S
075F  61 69 73 68 6F 00 0A 20              aisho..
0767  54 72 69 75 6D 70 68 00              Triumph.
076F  0A 20 49 73 70 00                    . Isp.

0775  E9                      jp     (hl)

*************************************************** MC SET MODE
0776  FE03                    cp     03          Mode > 2 ?
0778  D0                      ret    nc          ja ⇨
0779  F3                      di
077A  D9                      exx
077B  CB89                    res    1,c         Mode Bits
077D  CB81                    res    0,c         rücksetzen
077F  B1                      or     c
0780  4F                      ld     c,a         neuen Mode
0781  ED49                    out    (c),c       setzen
0783  FB                      ei
0784  D9                      exx
0785  C9                      ret

*************************************************** MC CLEAR INKS
0786  C5                      push   bc
0787  D5                      push   de
0788  01107F                  ld     bc,7F10
078B  CDAB07                  call   07AB        Farbe ausgeben
078E  0E00                    ld     c,00
0790  CDAB07                  call   07AB        Farbe ausgeben
0793  1B                      dec    de
0794  20FA                    jr     nz,0790
0796  D1                      pop    de
0797  C1                      pop    bc
0798  C9                      ret

*************************************************** MC SET INKS
0799  C5                      push   bc
079A  D5                      push   de
079B  01107F                  ld     bc,7F10     Border Farbe
079E  CDAB07                  call   07AB        Farbe ausgeben
07A1  0E00                    ld     c,00        Adr. Ink 0
07A3  CDAB07                  call   07AB        Farbe ausgeben
07A6  20FB                    jr     nz,07A3     alle Farbspeicher laden
07A8  D1                      pop    de
07A9  C1                      pop    bc
07AA  C9                      ret
```

```
************************************************ Farbe ausgeben
07AB  ED49          out   (c),c         Palette Pointer
07AD  1A            ld    a,(de)
07AE  13            inc   de
07AF  E61F          and   1F
07B1  F640          or    40
07B3  ED79          out   (c),a         Farbe
07B5  0C            inc   c
07B6  79            ld    a,c
07B7  FE10          cp    10
07B9  C9            ret

********************************************** MC WAIT FLYBACK
07BA  F5            push  af
07BB  C5            push  bc
07BC  06F5          ld    b,F5          Port B
07BE  ED78          in    a,(c)
07C0  1F            rra                 VSYNC ?
07C1  30FB          jr    nc,07BE       nein ⇒ warten
07C3  C1            pop   bc
07C4  F1            pop   af
07C5  C9            ret

********************************************* MC SCREEN OFFSET
07C6  C5            push  bc
07C7  0F            rrca
07C8  0F            rrca
07C9  E630          and   30
07CB  4F            ld    c,a
07CC  7C            ld    a,h
07CD  1F            rra
07CE  E603          and   03
07D0  B1            or    c
07D1  010CBC        ld    bc,BC0C
07D4  ED49          out   (c),c         Video Contr Reg 12
07D6  04            inc   b
07D7  ED79          out   (c),a         Bildsch. Start Hi
07D9  05            dec   b
07DA  0C            inc   c
07DB  ED49          out   (c),c         Reg 13
07DD  04            inc   b
07DE  7C            ld    a,h
07DF  1F            rra
07E0  7D            ld    a,l
07E1  1F            rra
07E2  ED79          out   (c),a         Bildsch. Start Lo
07E4  C1            pop   bc
07E5  C9            ret
```

```
******************************************* MC RESET PRINTER
07E6  21EC07     ld    hl,07EC      Restore Printer Indirection
07E9  C38A0A     jp    0A8A         Move (hl+3)⇒((hl+1)),cnt=(hl)

07EC  03         db    03           3 Bytes
07ED  F1BD       dw    BDF1         Zieladresse
07EF  C3F807     jp    07F8         MC WAIT PRINTER

********************************************* MC PRINT CHAR
07F2  C5         push  bc
07F3  CDF1BD     call  BDF1         MC WAIT PRINTER
07F6  C1         pop   bc
07F7  C9         ret

******************************************** MC WAIT PRINTER
07F8  013200     ld    bc,0032
07FB  CD1B08     call  081B         MC BUSY PRINTER
07FE  3007       jr    nc,0807      MC SEND PRINTER
0800  10F9       djnz  07FB
0802  0D         dec   c
0803  20F6       jr    nz,07FB
0805  B7         or    a
0806  C9         ret

******************************************** MC SEND PRINTER
0807  C5         push  bc
0808  06EF       ld    b,EF
080A  E67F       and   7F           Byte ohne Strobe
080C  ED79       out   (c),a        an Drucker
080E  F680       or    80
0810  F3         di
0811  ED79       out   (c),a        Strobe Ein
0813  E67F       and   7F
0815  FB         ei
0816  ED79       out   (c),a        Strobe Aus
0818  C1         pop   bc
0819  37         scf
081A  C9         ret

******************************************** MC BUSY PRINTER
081B  C5         push  bc
081C  4F         ld    c,a
081D  06F5       ld    b,F5         Port B
081F  ED78       in    a,(c)
0821  17         rla                Drucker Busy
0822  17         rla                ⇒ Carry
0823  79         ld    a,c
0824  C1         pop   bc
0825  C9         ret
```

```
******************************************** MC SOUND REGISTER
0826  F3          di
0827  06F4        ld    b,F4          Port A
0829  ED79        out   (c),a         Sound Reg#
082B  06F6        ld    b,F6          Port C
082D  ED78        in    a,(c)         Sound Chip
082F  F6C0        or    C0            auf Eingabe
0831  ED79        out   (c),a         & Strobe Ein
0833  E63F        and   3F
0835  ED79        out   (c),a         Strobe Aus
0837  06F4        ld    b,F4          Port A
0839  ED49        out   (c),c         Sound Daten
083B  06F6        ld    b,F6          Port C
083D  4F          ld    c,a
083E  F680        or    80
0840  ED79        out   (c),a         Daten
0842  ED49        out   (c),c         einlatchen
0844  FB          ei
0845  C9          ret

************************************************ Scan Keyboard
0846  010EF4      ld    bc,F40E       Port A
0849  ED49        out   (c),c         Sound Reg 14 (Keyb X Input)
084B  06F6        ld    b,F6          Port C
084D  ED78        in    a,(c)
084F  E630        and   30
0851  4F          ld    c,a
0852  F6C0        or    C0
0854  ED79        out   (c),a         Strobe Ein
0856  ED49        out   (c),c         Strobe Aus
0858  04          inc   b
0859  3E92        ld    a,92          Port A&B=Input
085B  ED79        out   (c),a         Control
085D  C5          push  bc
085E  CBF1        set   6,c
0860  06F6        ld    b,F6          Port C
0862  ED49        out   (c),c         Keyb Y Outp & X Inp
0864  06F4        ld    b,F4          Port A
0866  ED78        in    a,(c)         Daten (Keyb X Inp) ⇒ a
0868  46          ld    b,(hl)
0869  77          ld    (hl),a
086A  A0          and   b
086B  2F          cpl
086C  12          ld    (de),a
086D  23          inc   hl
086E  13          inc   de
086F  0C          inc   c             Keyb Y +1
0870  79          ld    a,c
0871  E60F        and   0F
0873  FE0A        cp    0A            alle Y-Leitungen bearbeitet ?
0875  20E9        jr    nz,0860       nein ⇒ nächste
0877  C1          pop   bc
0878  3E82        ld    a,82          Port A Output
087A  ED79        out   (c),a         Control
087C  05          dec   b
```

```
087D  ED49        out   (c),c        Port C
087F  C9          ret

0880  C7          rst   0
0881  C7          rst   0
0882  C7          rst   0
0883  C7          rst   0
0884  C7          rst   0
0885  C7          rst   0
0886  C7          rst   0
0887  C7          rst   0
```

2.5.3 JUMP RESTORE (JRE)

Dieses Pack dient ausschließlich dazu, die *MAIN JUMP*-Adressen wieder auf ihre Default-Werte zu setzen.
Dabei wird bei den *FIRM JUMPS* ein RST1 vorangestellt, bei den *ARITHMETIK JUMPS* ein RST5

Wenn Sie der Meinung sind, allzu viele Vektoren verbogen zu haben, ziehen Sie einfach die 'Notbremse', indem Sie diesen *JUMP RESTORE* anspringen.
Das ist auch ratsam, wenn Sie ein Programm verlassen, in dem Sie dem Betriebssystem eigene Routinen 'untergejubelt' haben.

```
********************************************* JUMP RESTORE
0888  11AC08     ld    de,08AC    Main Jump Adr.
088B  2100BB     ld    hl,BB00
088E  01CFBF     ld    bc,BFCF    RST1 vorsetzen
0891  CD9708     call  0897
0894  01EF30     ld    bc,30EF    RST5 vorsetzen
0897  71         ld    (hl),c
0898  23         inc   hl
0899  1A         ld    a,(de)
089A  77         ld    (hl),a
089B  13         inc   de
089C  23         inc   hl
089D  EB         ex    de,hl
089E  79         ld    a,c
089F  2F         cpl
08A0  07         rlca
08A1  07         rlca
08A2  E680       and   80
08A4  B6         or    (hl)
08A5  EB         ex    de,hl
08A6  77         ld    (hl),a
08A7  13         inc   de
08A8  23         inc   hl
08A9  10EC       djnz  0897
08AB  C9         ret

********************************************* Main Jump Adr.
08AC  E019       dw    19E0       KM INITIALISE
08AE  1E1A       dw    1A1E       KM RESET
08B0  3C1A       dw    1A3C       KM WAIT CHAR
08B2  421A       dw    1A42       KM READ CHAR
08B4  771A       dw    1A77       KM CHAR RETURN
08B6  BD1A       dw    1ABD       KM SET EXPAND
08B8  2E1B       dw    1B2E       KM GET EXPAND
08BA  7B1A       dw    1A7B       KM EXP BUFFER
08BC  561B       dw    1B56       KM WAIT KEY
08BE  5C1B       dw    1B5C       KM READ KEY
08C0  BD1C       dw    1CBD       KM TEST KEY
08C2  B31B       dw    1BB3       KM GET STATE
08C4  5C1C       dw    1C5C       KM GET JOYSTICK
08C6  521D       dw    1D52       KM SET TRANSLATE
08C8  3E1D       dw    1D3E       KM GET TRANSLATE
08CA  571D       dw    1D57       KM SET SHIFT
08CC  431D       dw    1D43       KM GET SHIFT
08CE  5C1D       dw    1D5C       KM SET CONTROL
08D0  481D       dw    1D48       KM GET CONTROL
08D2  AB1C       dw    1CAB       KM SET REPEAT
08D4  A61C       dw    1CA6       KM GET REPEAT
08D6  6D1C       dw    1C6D       KM SET DELAY
08D8  691C       dw    1C69       KM GET DELAY
08DA  711C       dw    1C71       KM ARM BREAK
08DC  821C       dw    1C82       KM DISARM BREAK
08DE  901C       dw    1C90       KM BREAK EVENT
08E0  7810       dw    1078       TXT INITIALISE
```

```
08E2  8810    dw   1088    TXT RESET
08E4  5114    dw   1451    TXT VDU ENABLE
08E6  4B14    dw   144B    TXT VDU DISABLE
08E8  0014    dw   1400    TXT OUTPUT
08EA  3413    dw   1334    TXT WR CHAR
08EC  AB13    dw   13AB    TXT RD CHAR
08EE  A713    dw   13A7    TXT SET GRAPHIC
08F0  0C12    dw   120C    TXT WIN ENABLE
08F2  5612    dw   1256    TXT GET WINDOW
08F4  4015    dw   1540    TXT CLEAR WINDOW
08F6  5E11    dw   115E    TXT SET COLUMN
08F8  6911    dw   1169    TXT SET ROW
08FA  7411    dw   1174    TXT SET CURSOR
08FC  8011    dw   1180    TXT GET CURSOR
08FE  8912    dw   1289    TXT CUR ENABLE
0900  9A12    dw   129A    TXT CUR DISABLE
0902  7912    dw   1279    TXT CUR ON
0904  8112    dw   1281    TXT CUR OFF
0906  CE11    dw   11CE    TXT VALIDATE
0908  6812    dw   1268    TXT PLACE/REMOVE CURSOR
090A  6812    dw   1268    TXT PLACE/REMOVE CURSOR
090C  A912    dw   12A9    TXT SET PEN
090E  BD12    dw   12BD    TXT GET PEN
0910  AE12    dw   12AE    TXT SET PAPER
0912  C312    dw   12C3    TXT GET PAPER
0914  C912    dw   12C9    TXT INVERSE
0916  7A13    dw   137A    TXT SET BACK
0918  8713    dw   1387    TXT GET BACK
091A  D312    dw   12D3    TXT GET MATRIX
091C  F112    dw   12F1    TXT SET MATRIX
091E  FD12    dw   12FD    TXT SET M TABLE
0920  2A13    dw   132A    TXT GET M TABLE
0922  CB14    dw   14CB    TXT GET CONTROLS
0924  E810    dw   10E8    TXT STR SELECT
0926  0711    dw   1107    TXT SWAP STREAMS
0928  B015    dw   15B0    GRA INITIALISE
092A  DF15    dw   15DF    GRA RESET
092C  F415    dw   15F4    GRA MOVE ABSOLUTE
092E  F115    dw   15F1    GRA MOVE RELATIVE
0930  FC15    dw   15FC    GRA ASK CURSOR
0932  0416    dw   1604    GRA SET ORIGIN
0934  1216    dw   1612    GRA GET ORIGIN
0936  3417    dw   1734    GRA WIN WIDTH
0938  7917    dw   1779    GRA WIN HEIGHT
093A  A617    dw   17A6    GRA GET W WIDTH
093C  BC17    dw   17BC    GRA GET W HEIGHT
093E  C517    dw   17C5    GRA CLEAR WINDOW
0940  F617    dw   17F6    GRA SET PEN
0942  0418    dw   1804    GRA GET PEN
0944  FD17    dw   17FD    GRA SET PAPER
0946  0A18    dw   180A    GRA GET PAPER
0948  1318    dw   1813    GRA PLOT ABSOLUTE
094A  1018    dw   1810    GRA PLOT RELATIVE
094C  2718    dw   1827    GRA TEST ABSOLUTE
094E  2418    dw   1824    GRA TEST RELATIVE
```

```
0950  3918    dw  1839   GRA LINE ABSOLUTE
0952  3618    dw  1836   GRA LINE RELATIVE
0954  4519    dw  1945   GRA WR CHAR
0956  A00A    dw  0AA0   SCR INITIALISE
0958  B10A    dw  0AB1   SCR RESET
095A  3C0B    dw  0B3C   SCR SET OFFSET
095C  450B    dw  0B45   SCR SET BASE
095E  500B    dw  0B50   SCR GET LOCATION
0960  CA0A    dw  0ACA   SCR SET MODE
0962  EC0A    dw  0AEC   SCR GET MODE
0964  F70A    dw  0AF7   SCR CLEAR
0966  570B    dw  0B57   SCR CHAR LIMITS
0968  640B    dw  0B64   SCR CHAR POSITION
096A  A90B    dw  0BA9   SCR DOT POSITION
096C  F90B    dw  0BF9   SCR NEXT BYTE
096E  050C    dw  0C05   SCR PREV BYTE
0970  130C    dw  0C13   SCR NEXT LINE
0972  2D0C    dw  0C2D   SCR PREV LINE
0974  860C    dw  0C86   SCR INK ENCODE
0976  A00C    dw  0CA0   SCR INK DECODE
0978  EC0C    dw  0CEC   SCR SET INK
097A  140D    dw  0D14   SCR GET INK
097C  F10C    dw  0CF1   SCR SET BORDER
097E  190D    dw  0D19   SCR GET BORDER
0980  E40C    dw  0CE4   SCR SET FLASHING
0982  E80C    dw  0CE8   SCR GET FLASHING
0984  B30D    dw  0DB3   SCR FILL BOX
0986  B70D    dw  0DB7   SCR FLOOD BOX
0988  DF0D    dw  0DDF   SCR CHAR INVERT
098A  FA0D    dw  0DFA   SCR HW ROLL
098C  3E0E    dw  0E3E   SCR SW ROLL
098E  F30E    dw  0EF3   SCR UNPACK
0990  490F    dw  0F49   SCR REPACK
0992  490C    dw  0C49   SCR ACCESS
0994  6B0C    dw  0C6B   SCR PIXELS
0996  C40F    dw  0FC4   SCR HORIZONTAL
0998  2F10    dw  102F   SCR VERTICAL
099A  7023    dw  2370   CAS INITIALISE
099C  7F23    dw  237F   CAS SET SPEED
099E  8E23    dw  238E   CAS NOISY
09A0  4B2A    dw  2A4B   CAS START MOTOR
09A2  4F2A    dw  2A4F   CAS STOP MOTOR
09A4  512A    dw  2A51   CAS RESTORE MOTOR
09A6  9223    dw  2392   CAS IN OPEN
09A8  FC23    dw  23FC   CAS IN CLOSE
09AA  0124    dw  2401   CAS IN ABANDON
09AC  3524    dw  2435   CAS IN CHAR
09AE  AB24    dw  24AB   CAS IN DIRECT
09B0  9A24    dw  249A   CAS RETURN
09B2  9624    dw  2496   CAS TEST EOF
09B4  AB23    dw  23AB   CAS OUT OPEN
09B6  1524    dw  2415   CAS OUT CLOSE
09B8  2E24    dw  242E   CAS OUT ABANDON
09BA  5B24    dw  245B   CAS OUT CHAR
09BC  EA24    dw  24EA   CAS OUT DIRECT
```

```
09BE  2825    dw  2528   CAS CATALOG
09C0  3F28    dw  283F   CAS WRITE
09C2  3628    dw  2836   CAS READ
09C4  5128    dw  2851   CAS CHECK
09C6  681E    dw  1E68   SOUND RESET
09C8  9F1F    dw  1F9F   SOUND QUEUE
09CA  6C20    dw  206C   SOUND CHECK
09CC  8920    dw  2089   SOUND ARM EVENT
09CE  4A20    dw  204A   SOUND RELEASE
09D0  CB1E    dw  1ECB   SOUND HOLD
09D2  E61E    dw  1EE6   SOUND CONTINUE
09D4  3823    dw  2338   SOUND AMPL ENVELOPE
09D6  3D23    dw  233D   SOUND TONE ENVELOPE
09D8  4923    dw  2349   SOUND A ADDRESS
09DA  4E23    dw  234E   SOUND T ADDRESS
09DC  5C00    dw  005C   KL CHOKE OFF
09DE  2903    dw  0329   KL ROM WALK
09E0  3203    dw  0332   KL INIT BACK
09E2  A102    dw  02A1   KL LOG EXT
09E4  B202    dw  02B2   KL FIND COMMAND
09E6  6301    dw  0163   KL NEW FRAME FLY
09E8  6A01    dw  016A   KL ADD FRAME FLY
09EA  7001    dw  0170   KL DEL FRAME FLY
09EC  7601    dw  0176   KL NEW FAST TICKER
09EE  7D01    dw  017D   KL ADD FAST TICKER
09F0  8301    dw  0183   Delete Fast Ticker
09F2  B301    dw  01B3   KL ADD TICKER
09F4  C501    dw  01C5   Delete Ticker
09F6  D201    dw  01D2   KL INIT EVENT
09F8  E201    dw  01E2   KL EVENT
09FA  2802    dw  0228   KL SYNC RESET
09FC  8502    dw  0285   KL DEL SYNCHRONOUS
09FE  5602    dw  0256   KL NEXT SYNC
0A00  1A02    dw  021A   KL DO SYNC
0A02  7702    dw  0277   KL DONE SYNC
0A04  9502    dw  0295   KL EVENT DISABLE
0A06  9B02    dw  029B   KL EVENT ENABLE
0A08  8E02    dw  028E   KL DISARM EVENT
0A0A  9900    dw  0099   KL TIME PLEASE
0A0C  A300    dw  00A3   KL TIME SET
0A0E  DC05    dw  05DC   MC BOOT PROGRAM
0A10  0B06    dw  060B   MC START PROGRAM
0A12  BA07    dw  07BA   MC WAIT FLYBACK
0A14  7607    dw  0776   MC SET MODE
0A16  C607    dw  07C6   MC SCREEN OFFSET
0A18  8607    dw  0786   MC CLEAR INKS
0A1A  9907    dw  0799   MC SET INKS
0A1C  E607    dw  07E6   MC RESET PRINTER
0A1E  F207    dw  07F2   MC PRINT CHAR
0A20  1B08    dw  081B   MC BUSY PRINTER
0A22  0708    dw  0807   MC SEND PRINTER
0A24  2608    dw  0826   MC SOUND REGISTER
0A26  8808    dw  0888   JUMP RESTORE
```

```
*********************************************** Basic Jump Adr.
0A28  982A     dw   2A98     EDIT
0A2A  182E     dw   2E18     FLO Var. (de) ⇨ (hl)
0A2C  292E     dw   2E29     FLO Int ⇨ Flo
0A2E  552E     dw   2E55     FLO 4-Byte ⇨ Flo
0A30  662E     dw   2E66     FLO Flo ⇨ Int
0A32  8E2E     dw   2E8E     FLO Flo ⇨ Int
0A34  A12E     dw   2EA1     FLO Fix
0A36  AC2E     dw   2EAC     FLO Int
0A38  B62E     dw   2EB6
0A3A  1D2F     dw   2F1D     FLO Zahl * 10↑a
0A3C  3F33     dw   333F     FLO Add
0A3E  3733     dw   3337     FLO Sub
0A40  3B33     dw   333B     FLO Sub
0A42  1534     dw   3415     FLO Mul
0A44  9E34     dw   349E     FLO Div
0A46  7835     dw   3578     FLO Zahl * 2↑a
0A48  9A35     dw   359A     FLO Cmp
0A4A  F835     dw   35F8     FLO +/-
0A4C  E835     dw   35E8     FLO Sgn
0A4E  AE31     dw   31AE     FLO Deg/Rad
0A50  A331     dw   31A3     FLO Pi
0A52  0A31     dw   310A     FLO Sqr
0A54  0D31     dw   310D     FLO Potenz
0A56  1430     dw   3014     FLO Log
0A58  0F30     dw   300F     FLO Log10
0A5A  9030     dw   3090     FLO Exp
0A5C  BC31     dw   31BC     FLO Sin
0A5E  B231     dw   31B2     FLO Cos
0A60  3132     dw   3231     FLO Tan
0A62  4132     dw   3241     FLO Atn
0A64  5E2E     dw   2E5E     FLO 4-Byte * 256 ⇨ Flo
0A66  942F     dw   2F94     FLO RND Init
0A68  A12F     dw   2FA1     FLO SET RANDOM SEED
0A6A  B72F     dw   2FB7     FLO RND
0A6C  E62F     dw   2FE6     FLO GET LAST RND
0A6E  0837     dw   3708
0A70  0E37     dw   370E
0A72  1537     dw   3715     INT Sgn in b übernehmen
0A74  2837     dw   3728     INT Add
0A76  3137     dw   3731     INT Sub
0A78  3037     dw   3730     INT Sub
0A7A  3937     dw   3739     INT Mul
0A7C  7A37     dw   377A     INT Div
0A7E  8137     dw   3781     INT Mod
0A80  5037     dw   3750     INT Mul unsigned
0A82  8C37     dw   378C     hl/de ⇨ hl, Rest ⇨ de
0A84  E937     dw   37E9     INT Cmp
0A86  D437     dw   37D4     INT +/-
0A88  E037     dw   37E0     INT Sgn
```

```
***************************************** Move (hl+3)⇨((hl+1)),cnt=(hl)
0A8A  4E          ld    c,(hl)
0A8B  0600        ld    b,00
0A8D  23          inc   hl
0A8E  5E          ld    e,(hl)
0A8F  23          inc   hl
0A90  56          ld    d,(hl)
0A91  23          inc   hl
0A92  EDB0        ldir
0A94  C9          ret

0A95  C7          rst   0
0A96  C7          rst   0
0A97  C7          rst   0
0A98  C7          rst   0
0A99  C7          rst   0
0A9A  C7          rst   0
0A9B  C7          rst   0
0A9C  C7          rst   0
0A9D  C7          rst   0
0A9E  C7          rst   0
0A9F  C7          rst   0
```

2.5.4 SCREEN PACK (SCR)

Das *SCREEN PACK* ist dem *TEXT-* und *GRAPHICS PACK* untergeordnet. Es ist praktisch die Exekutive für die Beiden und damit für die unmittelbare Handhabung des Bildschirmes zuständig.

An Routinen wollen wir Ihnen vorstellen:

SCR NEXT BYTE und *SCR PREV BYTE* liefern in **hl** die Bildschirmadresse der nächsten / vorigen Byteposition zurück, wenn Sie vor dem Ansprung **hl** mit der alten Adresse versorgt haben. So überflüssig das aussieht, so praktisch ist es. Es ist nämlich, aufgrund der auf den Grafikbetrieb ausgerichteten Organisation des Bildschirmes, nicht einfach, die Byteposition zu ermitteln. Zudem ist die Distanz vom Modus abhängig. Beachten Sie, daß, wenn die nächste oder vorige Position nicht mehr innerhalb des Bildschirmes läge, die zurückgelieferte Adresse unsinnig ist. Sie liegt dann im Bereich der letzten 48 (für die Darstellung unbenutzten) Bytes des Videorams.

SCR NEXT LINE und *SCR PREV LINE* arbeiten analog, nur daß die Bildschirmadresse um eine ganze Zeile vor- oder zurückgerechnet wird. Auch hier ist die Adresse beim Verlassen des darstellbaren Bereiches ungültig.

SCR HW ROLL schiebt den Bildschirm um eine Zeile nach unten, wenn **b**=0 ist, und eine Zeile nach oben, wenn **b**<>0 ist.
a ist mit der Farbe zu versorgen, die die neue (leere) Zeile annehmen soll.

SCR SW ROLL verschiebt einen Bildschirmbereich. **a** und **b** sind wie oben zu versorgen. Zusätzlich muß **h** die Spaltennummer des linken Randes des zu verschiebenden Bereiches enthalten, **l** die oberste Zeile, **d** die rechte Spalte und **e** die unterste Zeile des Bereiches.
Beachten Sie, daß Spalte und Zeile 0 die linke obere Ecke des Bildschirmes darstellt. Achten Sie auch unbedingt selbst darauf, daß die übergebenen Parameter tatsächlich einen Bereich innerhalb des Videorams markieren.

```
************************************************** SCR INITIALISE
0AA0  114D10      ld     de,104D       Default Farben
0AA3  CD8607      call   0786          MC CLEAR INKS
0AA6  3EC0        ld     a,C0
0AA8  32CBB1      ld     (B1CB),a      (High Byte Screen Start)
0AAB  CDB10A      call   0AB1          SCR RESET
0AAE  C3F20A      jp     0AF2

**************************************************** SCR RESET
0AB1  AF          xor    a
0AB2  CD490C      call   0C49          SCR ACCESS
0AB5  21BE0A      ld     hl,0ABE       Restore SCR Indirections
0AB8  CD8A0A      call   0A8A          Move (hl+3)⇒((hl+1)),cnt=(hl)
0ABB  C3D20C      jp     0CD2          Reset Farben

0ABE  09          db     09            9 Bytes
0ABF  E5BD        dw     BDE5          Zieladresse
0AC1  C3820C      jp     0C82          SCR READ

0AC4  C3680C      jp     0C68          SCR WRITE

0AC7  C3F70A      jp     0AF7          SCR CLEAR

************************************************** SCR SET MODE
0ACA  E603        and    03
0ACC  FE03        cp     03
0ACE  D0          ret    nc
0ACF  F5          push   af
0AD0  CD4F0D      call   0D4F
0AD3  F1          pop    af
0AD4  5F          ld     e,a
0AD5  CDB710      call   10B7
0AD8  F5          push   af
0AD9  CDD615      call   15D6
0ADC  E5          push   hl
0ADD  7B          ld     a,e
0ADE  CD110B      call   0B11          Bit Masken laden
0AE1  CDEBBD      call   BDEB          SCR MODE CLEAR
0AE4  E1          pop    hl
0AE5  CDB615      call   15B6
0AE8  F1          pop    af
0AE9  C3D510      jp     10D5

************************************************* SCR GET MODE
0AEC  3AC8B1      ld     a,(B1C8)      (curr. Screen Mode)
0AEF  FE01        cp     01
0AF1  C9          ret

0AF2  3E01        ld     a,01
0AF4  CD110B      call   0B11          Bit Masken laden
```

```
******************************************** SCR MODE CLEAR
0AF7  CD4F0D      call   0D4F
0AFA  210000      ld     hl,0000
0AFD  CD3C0B      call   0B3C       SCR SET OFFSET
0B00  2ACAB1      ld     hl,(B1CA)  (Adr. Screen Start)
0B03  2E00        ld     l,00
0B05  54          ld     d,h        hl=Basis Adresse
0B06  1E01        ld     e,01       de=Basis Adresse +1
0B08  01FF3F      ld     bc,3FFF    16k
0B0B  75          ld     (hl),l
0B0C  EDB0        ldir              Bildschirm löschen
0B0E  C33C0D      jp     0D3C

********************************************* Bit Masken laden
0B11  213A0B      ld     hl,0B3A    Bit Masken Mode 0
0B14  FE01        cp     01
0B16  3808        jr     c,0B20
0B18  21360B      ld     hl,0B36    Bit Masken Mode 1
0B1B  2803        jr     z,0B20
0B1D  212E0B      ld     hl,0B2E    Bit Masken Mode 2
0B20  11CFB1      ld     de,B1CF    Bit Masken abh. v. Mode
0B23  010800      ld     bc,0008
0B26  EDB0        ldir
0B28  32C8B1      ld     (B1C8),a   (curr. Screen Mode)
0B2B  C37607      jp     0776       MC SET MODE

******************************************** Bit Masken Mode 2
0B2E  80 40 20 10 08 04 02 01

******************************************** Bit Masken Mode 1
0B36  88 44 22 11

******************************************** Bit Masken Mode 0
0B3A  AA 55

********************************************* SCR SET OFFSET

0B3C  7C          ld     a,h
0B3D  E607        and    07
0B3F  67          ld     h,a
0B40  22C9B1      ld     (B1C9),hl
0B43  1805        jr     0B4A

********************************************* SCR SET BASE
0B45  E6C0        and    C0
0B47  32CBB1      ld     (B1CB),a   (High Byte Screen Start)
0B4A  CD500B      call   0B50       SCR GET LOCATION
0B4D  C3C607      jp     07C6       MC SCREEN OFFSET

******************************************** SCR GET LOCATION
0B50  2AC9B1      ld     hl,(B1C9)
0B53  3ACBB1      ld     a,(B1CB)   (High Byte Screen Start)
0B56  C9          ret
```

```
********************************************** SCR CHAR LIMITS
0B57  CDEC0A      call   0AEC        SCR GET MODE
0B5A  011813      ld     bc,1318
0B5D  D8          ret    c
0B5E  0627        ld     b,27
0B60  C8          ret    z
0B61  064F        ld     b,4F
0B63  C9          ret

********************************************* SCR CHAR POSITION
0B64  D5          push   de
0B65  CDEC0A      call   0AEC        SCR GET MODE
0B68  0604        ld     b,04
0B6A  3805        jr     c,0B71
0B6C  0602        ld     b,02
0B6E  2801        jr     z,0B71
0B70  05          dec    b
0B71  C5          push   bc
0B72  5C          ld     e,h
0B73  1600        ld     d,00
0B75  62          ld     h,d
0B76  D5          push   de
0B77  54          ld     d,h
0B78  5D          ld     e,l
0B79  29          add    hl,hl
0B7A  29          add    hl,hl
0B7B  19          add    hl,de
0B7C  29          add    hl,hl
0B7D  29          add    hl,hl
0B7E  29          add    hl,hl
0B7F  29          add    hl,hl
0B80  D1          pop    de
0B81  19          add    hl,de
0B82  10FD        djnz   0B81
0B84  ED5BC9B1    ld     de,(B1C9)
0B88  19          add    hl,de
0B89  7C          ld     a,h
0B8A  E607        and    07
0B8C  67          ld     h,a
0B8D  3ACBB1      ld     a,(B1CB)    (High Byte Screen Start)
0B90  84          add    a,h
0B91  67          ld     h,a
0B92  C1          pop    bc
0B93  D1          pop    de
0B94  C9          ret

0B95  7B          ld     a,e
0B96  95          sub    l
0B97  3C          inc    a
0B98  87          add    a,a
0B99  87          add    a,a
0B9A  87          add    a,a
0B9B  5F          ld     e,a
0B9C  7A          ld     a,d
0B9D  94          sub    h
```

```
0B9E  3C             inc   a
0B9F  57             ld    d,a
0BA0  CD640B         call  0B64          SCR CHAR POSITION
0BA3  AF             xor   a
0BA4  82             add   a,d
0BA5  10FD           djnz  0BA4
0BA7  57             ld    d,a
0BA8  C9             ret

********************************************* SCR DOT POSITION
0BA9  D5             push  de
0BAA  EB             ex    de,hl
0BAB  21C700         ld    hl,00C7
0BAE  B7             or    a
0BAF  ED52           sbc   hl,de
0BB1  7D             ld    a,l
0BB2  E607           and   07
0BB4  87             add   a,a
0BB5  87             add   a,a
0BB6  87             add   a,a
0BB7  4F             ld    c,a
0BB8  7D             ld    a,l
0BB9  E6F8           and   F8
0BBB  6F             ld    l,a
0BBC  54             ld    d,h
0BBD  5D             ld    e,l
0BBE  29             add   hl,hl
0BBF  29             add   hl,hl
0BC0  19             add   hl,de
0BC1  29             add   hl,hl
0BC2  D1             pop   de
0BC3  CDEC0A         call  0AEC          SCR GET MODE
0BC6  0601           ld    b,01
0BC8  3806           jr    c,0BD0
0BCA  0603           ld    b,03
0BCC  2802           jr    z,0BD0
0BCE  0607           ld    b,07
0BD0  78             ld    a,b
0BD1  A3             and   e
0BD2  F5             push  af
0BD3  78             ld    a,b
0BD4  0F             rrca
0BD5  CB3A           srl   d
0BD7  CB1B           rr    e
0BD9  0F             rrca
0BDA  38F9           jr    c,0BD5
0BDC  19             add   hl,de
0BDD  ED5BC9B1       ld    de,(B1C9)
0BE1  19             add   hl,de
0BE2  7C             ld    a,h
0BE3  E607           and   07
0BE5  67             ld    h,a
0BE6  3ACBB1         ld    a,(B1CB)      (High Byte Screen Start)
0BE9  84             add   a,h
0BEA  81             add   a,c
```

```
0BEB  67          ld    h,a
0BEC  F1          pop   af
0BED  E5          push  hl
0BEE  1600        ld    d,00
0BF0  5F          ld    e,a
0BF1  21CFB1      ld    hl,B1CF        Bit Masken abh. v. Mode
0BF4  19          add   hl,de
0BF5  4E          ld    c,(hl)
0BF6  EB          ex    de,hl
0BF7  E1          pop   hl
0BF8  C9          ret

*********************************************** SCR NEXT BYTE
0BF9  2C          inc   l
0BFA  C0          ret   nz
0BFB  24          inc   h
0BFC  7C          ld    a,h
0BFD  E607        and   07
0BFF  C0          ret   nz
0C00  7C          ld    a,h
0C01  D608        sub   08
0C03  67          ld    h,a
0C04  C9          ret

*********************************************** SCR PREV BYTE
0C05  7D          ld    a,l
0C06  2D          dec   l
0C07  B7          or    a
0C08  C0          ret   nz
0C09  7C          ld    a,h
0C0A  25          dec   h
0C0B  E607        and   07
0C0D  C0          ret   nz
0C0E  7C          ld    a,h
0C0F  C608        add   a,08
0C11  67          ld    h,a
0C12  C9          ret

************************************************ SCR NEXT LINE
0C13  7C          ld    a,h
0C14  C608        add   a,08
0C16  67          ld    h,a
0C17  E638        and   38
0C19  C0          ret   nz
0C1A  7C          ld    a,h
0C1B  D640        sub   40
0C1D  67          ld    h,a
0C1E  7D          ld    a,l
0C1F  C650        add   a,50
0C21  6F          ld    l,a
0C22  D0          ret   nc
0C23  24          inc   h
0C24  7C          ld    a,h
0C25  E607        and   07
0C27  C0          ret   nz
```

```
0C28  7C          ld    a,h
0C29  D608        sub   08
0C2B  67          ld    h,a
0C2C  C9          ret

************************************************** SCR PREV LINE
0C2D  7C          ld    a,h
0C2E  D608        sub   08
0C30  67          ld    h,a
0C31  E638        and   38
0C33  FE38        cp    38
0C35  C0          ret   nz
0C36  7C          ld    a,h
0C37  C640        add   a,40
0C39  67          ld    h,a
0C3A  7D          ld    a,l
0C3B  D650        sub   50
0C3D  6F          ld    l,a
0C3E  D0          ret   nc
0C3F  7C          ld    a,h
0C40  25          dec   h
0C41  E607        and   07
0C43  C0          ret   nz
0C44  7C          ld    a,h
0C45  C608        add   a,08
0C47  67          ld    h,a
0C48  C9          ret

*************************************************** SCR ACCESS
0C49  E603        and   03
0C4B  216B0C      ld    hl,0C6B     SCR PIXELS (FORCE Mode)
0C4E  280F        jr    z,0C5F
0C50  FE02        cp    02
0C52  21720C      ld    hl,0C72     XOR Mode
0C55  3808        jr    c,0C5F
0C57  21770C      ld    hl,0C77     AND Mode
0C5A  2803        jr    z,0C5F
0C5C  217D0C      ld    hl,0C7D     OR Mode
0C5F  3EC3        ld    a,C3        jp
0C61  32CCB1      ld    (B1CC),a    (SCR Write Indirection)
0C64  22CDB1      ld    (B1CD),hl
0C67  C9          ret

**************************************************** SCR WRITE
0C68  C3CCB1      jp    B1CC        SCR Write Indirection

******************************************** SCR PIXELS (FORCE Mode)
0C6B  7E          ld    a,(hl)
0C6C  A8          xor   b
0C6D  B1          or    c
0C6E  A9          xor   c
0C6F  A8          xor   b
0C70  77          ld    (hl),a
0C71  C9          ret
```

```
************************************************************ XOR Mode
0C72  78              ld      a,b
0C73  A1              and     c
0C74  AE              xor     (hl)
0C75  77              ld      (hl),a
0C76  C9              ret

************************************************************ AND Mode
0C77  79              ld      a,c
0C78  2F              cpl
0C79  B0              or      b
0C7A  A6              and     (hl)
0C7B  77              ld      (hl),a
0C7C  C9              ret

************************************************************* OR Mode
0C7D  78              ld      a,b
0C7E  A1              and     c
0C7F  B6              or      (hl)
0C80  77              ld      (hl),a
0C81  C9              ret

************************************************************ SCR READ
0C82  7E              ld      a,(hl)
0C83  C3AC0C          jp      0CAC

******************************************************** SCR INK ENCODE
0C86  C5              push    bc
0C87  D5              push    de
0C88  CDC20C          call    0CC2
0C8B  5F              ld      e,a
0C8C  0608            ld      b,08
0C8E  3ACFB1          ld      a,(B1CF)          (Bit Masken abh. v. Mode)
0C91  4F              ld      c,a
0C92  CB0B            rrc     e
0C94  17              rla
0C95  CB09            rrc     c
0C97  3802            jr      c,0C9B
0C99  CB03            rlc     e
0C9B  10F5            djnz    0C92
0C9D  D1              pop     de
0C9E  C1              pop     bc
0C9F  C9              ret

******************************************************** SCR INK DECODE
0CA0  C5              push    bc
0CA1  47              ld      b,a
0CA2  3ACFB1          ld      a,(B1CF)          (Bit Masken abh. v. Mode)
0CA5  4F              ld      c,a
0CA6  78              ld      a,b
0CA7  CDAC0C          call    0CAC
0CAA  C1              pop     bc
0CAB  C9              ret

0CAC  D5              push    de
```

```
0CAD 110800      ld    de,0008
0CB0 0F          rrca
0CB1 CB12        rl    d
0CB3 CB09        rrc   c
0CB5 3802        jr    c,0CB9
0CB7 CB1A        rr    d
0CB9 1D          dec   e
0CBA 20F4        jr    nz,0CB0
0CBC 7A          ld    a,d
0CBD CDC20C      call  0CC2
0CC0 D1          pop   de
0CC1 C9          ret

0CC2 57          ld    d,a
0CC3 CDEC0A      call  0AEC         SCR GET MODE
0CC6 7A          ld    a,d
0CC7 D0          ret   nc
0CC8 0F          rrca
0CC9 0F          rrca
0CCA CE00        adc   a,00
0CCC 0F          rrca
0CCD 9F          sbc   a,a
0CCE E606        and   06
0CD0 AA          xor   d
0CD1 C9          ret

************************************************ Reset Farben
0CD2 214D10      ld    hl,104D      Default Farben
0CD5 11D9B1      ld    de,B1D9      Farbspeicher 2. Farben
0CD8 012200      ld    bc,0022
0CDB EDB0        ldir
0CDD AF          xor   a
0CDE 32FBB1      ld    (B1FB),a     (Flag lfd. Farbsatz)
0CE1 210A0A      ld    hl,0A0A      (0A0A)=9900

********************************************* SCR SET FLASHING
0CE4 22D7B1      ld    (B1D7),hl    (Flash Periods)
0CE7 C9          ret

********************************************* SCR GET FLASHING
0CE8 2AD7B1      ld    hl,(B1D7)    (Flash Periods)
0CEB C9          ret

************************************************ SCR SET INK
0CEC E60F        and   0F
0CEE 3C          inc   a
0CEF 1801        jr    0CF2         Set Colour
```

```
*********************************************** SCR SET BORDER
0CF1  AF          xor   a

**************************************************** Set Colour
0CF2  5F          ld    e,a
0CF3  78          ld    a,b
0CF4  CD0A0D      call  0D0A      Farbmatrix Eintrag holen
0CF7  46          ld    b,(hl)
0CF8  79          ld    a,c
0CF9  CD0A0D      call  0D0A      Farbmatrix Eintrag holen
0CFC  4E          ld    c,(hl)
0CFD  7B          ld    a,e
0CFE  CD2F0D      call  0D2F      Ink Adr. holen
0D01  71          ld    (hl),c
0D02  EB          ex    de,hl
0D03  70          ld    (hl),b
0D04  3EFF        ld    a,FF
0D06  32FCB1      ld    (B1FC),a
0D09  C9          ret

******************************************** Farbmatrix Eintrag holen
0D0A  E61F        and   1F
0D0C  C693        add   a,93
0D0E  6F          ld    l,a
0D0F  CE0D        adc   a,0D
0D11  95          sub   l
0D12  67          ld    h,a
0D13  C9          ret

************************************************* SCR GET INK
0D14  E60F        and   0F
0D16  3C          inc   a
0D17  1801        jr    0D1A      Get Colour

*********************************************** SCR GET BORDER
0D19  AF          xor   a

**************************************************** Get Colour
0D1A  CD2F0D      call  0D2F      Ink Adr. holen
0D1D  1A          ld    a,(de)
0D1E  5E          ld    e,(hl)
0D1F  CD240D      call  0D24
0D22  41          ld    b,c
0D23  7B          ld    a,e
0D24  0E00        ld    c,00
0D26  21930D      ld    hl,0D93   Farbmatrix
0D29  BE          cp    (hl)
0D2A  C8          ret   z
0D2B  23          inc   hl
0D2C  0C          inc   c
0D2D  18FA        jr    0D29
```

```
************************************************** Ink Adr. holen
0D2F  5F          ld    e,a
0D30  1600        ld    d,00
0D32  21EAB1      ld    hl,B1EA       Farbspeicher 1. Farben
0D35  19          add   hl,de
0D36  EB          ex    de,hl
0D37  21EFFF      ld    hl,FFEF
0D3A  19          add   hl,de
0D3B  C9          ret

0D3C  21FEB1      ld    hl,B1FE       Event Block: Set Inks
0D3F  E5          push  hl
0D40  CD7001      call  0170          KL DEL FRAME FLY
0D43  CD6D0D      call  0D6D          Flash Inks
0D46  115B0D      ld    de,0D5B       Set Inks on Frame Fly
0D49  0681        ld    b,81
0D4B  E1          pop   hl
0D4C  C36301      jp    0163          KL NEW FRAME FLY

0D4F  21FEB1      ld    hl,B1FE       Event Block: Set Inks
0D52  CD7001      call  0170          KL DEL FRAME FLY
0D55  CD810D      call  0D81          Params d. lfd Farbsatz holen
0D58  C38607      jp    0786          MC CLEAR INKS

********************************************** Set Inks on Frame Fly
0D5B  21FDB1      ld    hl,B1FD       curr. Flash Period
0D5E  35          dec   (hl)
0D5F  280C        jr    z,0D6D        Flash Inks
0D61  2B          dec   hl
0D62  7E          ld    a,(hl)
0D63  B7          or    a
0D64  C8          ret   z
0D65  CD810D      call  0D81          Params d. lfd Farbsatz holen
0D68  CD9907      call  0799          MC SET INKS
0D6B  180F        jr    0D7C

***************************************************** Flash Inks
0D6D  CD810D      call  0D81          Params d. lfd Farbsatz holen
0D70  32FDB1      ld    (B1FD),a      (curr. Flash Period)
0D73  CD9907      call  0799          MC SET INKS
0D76  21FBB1      ld    hl,B1FB       Flag lfd. Farbsatz
0D79  7E          ld    a,(hl)
0D7A  2F          cpl
0D7B  77          ld    (hl),a
0D7C  AF          xor   a
0D7D  32FCB1      ld    (B1FC),a
0D80  C9          ret

****************************************** Params d. lfd Farbsatz holen
0D81  11EAB1      ld    de,B1EA       Farbspeicher 1. Farben
0D84  3AFBB1      ld    a,(B1FB)      (Flag lfd. Farbsatz)
0D87  B7          or    a
0D88  3AD8B1      ld    a,(B1D8)      (Flash Period 1.Colour)
0D8B  C8          ret   z
0D8C  11D9B1      ld    de,B1D9       Farbspeicher 2. Farben
```

```
0D8F  3AD7B1     ld     a,(B1D7)     (Flash Periods)
0D92  C9         ret

***************************************************** Farbmatrix
0D93  14 04 15 1C 18 1D 0C 05
0D9B  0D 16 06 17 1E 00 1F 0E
0DA3  07 0F 12 02 13 1A 19 1B
0DAB  0A 03 0B 01 08 09 10 11

**************************************************** SCR FILL BOX

0DB3  4F         ld     c,a
0DB4  CD950B     call   0B95

*************************************************** SCR FLOOD BOX
0DB7  E5         push   hl
0DB8  7A         ld     a,d
0DB9  CDE80E     call   0EE8
0DBC  3009       jr     nc,0DC7
0DBE  42         ld     b,d
0DBF  71         ld     (hl),c
0DC0  CDF90B     call   0BF9         SCR NEXT BYTE
0DC3  10FA       djnz   0DBF
0DC5  1810       jr     0DD7
0DC7  C5         push   bc
0DC8  D5         push   de
0DC9  71         ld     (hl),c
0DCA  15         dec    d
0DCB  2808       jr     z,0DD5
0DCD  4A         ld     c,d
0DCE  0600       ld     b,00
0DD0  54         ld     d,h
0DD1  5D         ld     e,l
0DD2  13         inc    de
0DD3  EDB0       ldir
0DD5  D1         pop    de
0DD6  C1         pop    bc
0DD7  E1         pop    hl
0DD8  CD130C     call   0C13         SCR NEXT LINE
0DDB  1D         dec    e
0DDC  20D9       jr     nz,0DB7      SCR FLOOD BOX
0DDE  C9         ret

************************************************** SCR CHAR INVERT
0DDF  78         ld     a,b
0DE0  A9         xor    c
0DE1  4F         ld     c,a
0DE2  CD640B     call   0B64         SCR CHAR POSITION
0DE5  1608       ld     d,08
0DE7  E5         push   hl
0DE8  C5         push   bc
0DE9  7E         ld     a,(hl)
0DEA  A9         xor    c
0DEB  77         ld     (hl),a
0DEC  CDF90B     call   0BF9         SCR NEXT BYTE
```

```
0DEF  10F8        djnz  0DE9
0DF1  C1          pop   bc

********************************************* Farbspeicher adressieren
0DF2  E1          pop   hl
0DF3  CD130C      call  0C13        SCR NEXT LINE
0DF6  15          dec   d
0DF7  20EE        jr    nz,0DE7
0DF9  C9          ret

************************************************* SCR HW ROLL
0DFA  4F          ld    c,a
0DFB  C5          push  bc
0DFC  11D0FF      ld    de,FFD0
0DFF  0630        ld    b,30
0E01  CD240E      call  0E24
0E04  C1          pop   bc
0E05  CDBA07      call  07BA        MC WAIT FLYBACK
0E08  78          ld    a,b
0E09  B7          or    a
0E0A  200D        jr    nz,0E19
0E0C  11B0FF      ld    de,FFB0
0E0F  CD370E      call  0E37
0E12  110000      ld    de,0000
0E15  0620        ld    b,20
0E17  180B        jr    0E24
0E19  115000      ld    de,0050
0E1C  CD370E      call  0E37
0E1F  11B0FF      ld    de,FFB0
0E22  0620        ld    b,20
0E24  2AC9B1      ld    hl,(B1C9)
0E27  19          add   hl,de
0E28  7C          ld    a,h
0E29  E607        and   07
0E2B  67          ld    h,a
0E2C  3ACBB1      ld    a,(B1CB)    (High Byte Screen Start)
0E2F  84          add   a,h
0E30  67          ld    h,a
0E31  50          ld    d,b
0E32  1E08        ld    e,08
0E34  C3B70D      jp    0DB7        SCR FLOOD BOX

0E37  2AC9B1      ld    hl,(B1C9)
0E3A  19          add   hl,de
0E3B  C33C0B      jp    0B3C        SCR SET OFFSET

************************************************* SCR SW ROLL
0E3E  F5          push  af
0E3F  78          ld    a,b
0E40  B7          or    a
0E41  2830        jr    z,0E73
0E43  E5          push  hl
0E44  CD950B      call  0B95
0E47  E3          ex    (sp),hl
0E48  2C          inc   l
```

```
0E49  CD640B    call  0B64      SCR CHAR POSITION
0E4C  4A        ld    c,d
0E4D  7B        ld    a,e
0E4E  D608      sub   08
0E50  47        ld    b,a
0E51  2817      jr    z,0E6A
0E53  D1        pop   de
0E54  CDBA07    call  07BA      MC WAIT FLYBACK
0E57  C5        push  bc
0E58  E5        push  hl
0E59  D5        push  de
0E5A  CDA40E    call  0EA4
0E5D  E1        pop   hl
0E5E  CD130C    call  0C13      SCR NEXT LINE
0E61  EB        ex    de,hl
0E62  E1        pop   hl
0E63  CD130C    call  0C13      SCR NEXT LINE
0E66  C1        pop   bc
0E67  10EE      djnz  0E57
0E69  D5        push  de
0E6A  E1        pop   hl
0E6B  51        ld    d,c
0E6C  1E08      ld    e,08
0E6E  F1        pop   af
0E6F  4F        ld    c,a
0E70  C3B70D    jp    0DB7      SCR FLOOD BOX

0E73  E5        push  hl
0E74  D5        push  de
0E75  CD950B    call  0B95
0E78  4A        ld    c,d
0E79  7B        ld    a,e
0E7A  D608      sub   08
0E7C  47        ld    b,a
0E7D  D1        pop   de
0E7E  E3        ex    (sp),hl
0E7F  28E9      jr    z,0E6A
0E81  C5        push  bc
0E82  6B        ld    l,e
0E83  54        ld    d,h
0E84  1C        inc   e
0E85  CD640B    call  0B64      SCR CHAR POSITION
0E88  EB        ex    de,hl
0E89  CD640B    call  0B64      SCR CHAR POSITION
0E8C  C1        pop   bc
0E8D  CDBA07    call  07BA      MC WAIT FLYBACK
0E90  CD2D0C    call  0C2D      SCR PREV LINE
0E93  E5        push  hl
0E94  EB        ex    de,hl
0E95  CD2D0C    call  0C2D      SCR PREV LINE
0E98  E5        push  hl
0E99  C5        push  bc
0E9A  CDA40E    call  0EA4
0E9D  C1        pop   bc
0E9E  D1        pop   de
```

```
0E9F  E1        pop   hl
0EA0  10EE      djnz  0E90
0EA2  18C6      jr    0E6A
0EA4  0600      ld    b,00
0EA6  CDE60E    call  0EE6
0EA9  3816      jr    c,0EC1
0EAB  CDE60E    call  0EE6
0EAE  3025      jr    nc,0ED5
0EB0  C5        push  bc
0EB1  AF        xor   a
0EB2  95        sub   l
0EB3  4F        ld    c,a
0EB4  EDB0      ldir
0EB6  C1        pop   bc
0EB7  2F        cpl
0EB8  3C        inc   a
0EB9  81        add   a,c
0EBA  4F        ld    c,a
0EBB  7C        ld    a,h
0EBC  D608      sub   08
0EBE  67        ld    h,a
0EBF  1814      jr    0ED5
0EC1  CDE60E    call  0EE6
0EC4  3812      jr    c,0ED8
0EC6  C5        push  bc
0EC7  AF        xor   a
0EC8  93        sub   e
0EC9  4F        ld    c,a
0ECA  EDB0      ldir
0ECC  C1        pop   bc
0ECD  2F        cpl
0ECE  3C        inc   a
0ECF  81        add   a,c
0ED0  4F        ld    c,a
0ED1  7A        ld    a,d
0ED2  D608      sub   08
0ED4  57        ld    d,a
0ED5  EDB0      ldir
0ED7  C9        ret

0ED8  41        ld    b,c
0ED9  7E        ld    a,(hl)
0EDA  12        ld    (de),a
0EDB  CDF90B    call  0BF9          SCR NEXT BYTE
0EDE  EB        ex    de,hl
0EDF  CDF90B    call  0BF9          SCR NEXT BYTE
0EE2  EB        ex    de,hl
0EE3  10F4      djnz  0ED9
0EE5  C9        ret
```

```
0EE6  79          ld    a,c
0EE7  EB          ex    de,hl
0EE8  3D          dec   a
0EE9  85          add   a,l
0EEA  D0          ret   nc
0EEB  7C          ld    a,h
0EEC  E607        and   07
0EEE  EE07        xor   07
0EF0  C0          ret   nz
0EF1  37          scf
0EF2  C9          ret

*********************************************** SCR UNPACK
0EF3  CDEC0A      call  0AEC        SCR GET MODE
0EF6  0608        ld    b,08
0EF8  3831        jr    c,0F2B
0EFA  2806        jr    z,0F02
0EFC  010800      ld    bc,0008
0EFF  EDB0        ldir
0F01  C9          ret

0F02  4E          ld    c,(hl)
0F03  23          inc   hl
0F04  E5          push  hl
0F05  C5          push  bc
0F06  0604        ld    b,04
0F08  21CFB1      ld    hl,B1CF     Bit Masken abh. v. Mode
0F0B  AF          xor   a
0F0C  CB01        rlc   c
0F0E  3001        jr    nc,0F11
0F10  B6          or    (hl)
0F11  23          inc   hl
0F12  10F8        djnz  0F0C
0F14  12          ld    (de),a
0F15  13          inc   de
0F16  0604        ld    b,04
0F18  21CFB1      ld    hl,B1CF     Bit Masken abh. v. Mode
0F1B  AF          xor   a
0F1C  CB01        rlc   c
0F1E  3001        jr    nc,0F21
0F20  B6          or    (hl)
0F21  23          inc   hl
0F22  10F8        djnz  0F1C
0F24  12          ld    (de),a
0F25  13          inc   de
0F26  C1          pop   bc
0F27  E1          pop   hl
0F28  10D8        djnz  0F02
0F2A  C9          ret

0F2B  4E          ld    c,(hl)
0F2C  23          inc   hl
0F2D  E5          push  hl
0F2E  C5          push  bc
0F2F  0604        ld    b,04
```

```
0F31  AF          xor   a
0F32  21CFB1      ld    hl,B1CF      Bit Masken abh. v. Mode
0F35  CB01        rlc   c
0F37  3001        jr    nc,0F3A
0F39  7E          ld    a,(hl)
0F3A  23          inc   hl
0F3B  CB01        rlc   c
0F3D  3001        jr    nc,0F40
0F3F  B6          or    (hl)
0F40  12          ld    (de),a
0F41  13          inc   de
0F42  10ED        djnz  0F31
0F44  C1          pop   bc
0F45  E1          pop   hl
0F46  10E3        djnz  0F2B
0F48  C9          ret

************************************************* SCR REPACK
0F49  4F          ld    c,a
0F4A  CD640B      call  0B64         SCR CHAR POSITION
0F4D  CDEC0A      call  0AEC         SCR GET MODE
0F50  0608        ld    b,08
0F52  3845        jr    c,0F99
0F54  280B        jr    z,0F61
0F56  7E          ld    a,(hl)
0F57  A9          xor   c
0F58  2F          cpl
0F59  12          ld    (de),a
0F5A  13          inc   de
0F5B  CD130C      call  0C13         SCR NEXT LINE
0F5E  10F6        djnz  0F56
0F60  C9          ret

0F61  E5          push  hl
0F62  D5          push  de
0F63  E5          push  hl
0F64  7E          ld    a,(hl)
0F65  A9          xor   c
0F66  21CFB1      ld    hl,B1CF      Bit Masken abh. v. Mode
0F69  1604        ld    d,04
0F6B  F5          push  af
0F6C  A6          and   (hl)
0F6D  2001        jr    nz,0F70
0F6F  37          scf
0F70  CB13        rl    e
0F72  23          inc   hl
0F73  F1          pop   af
0F74  15          dec   d
0F75  20F4        jr    nz,0F6B
0F77  E1          pop   hl
0F78  CDF90B      call  0BF9         SCR NEXT BYTE
0F7B  7E          ld    a,(hl)
0F7C  A9          xor   c
0F7D  21CFB1      ld    hl,B1CF      Bit Masken abh. v. Mode
0F80  1604        ld    d,04
```

```
0F82  F5        push  af
0F83  A6        and   (hl)
0F84  2001      jr    nz,0F87
0F86  37        scf
0F87  CB13      rl    e
0F89  23        inc   hl
0F8A  F1        pop   af
0F8B  15        dec   d
0F8C  20F4      jr    nz,0F82
0F8E  E1        pop   hl
0F8F  73        ld    (hl),e
0F90  EB        ex    de,hl
0F91  13        inc   de
0F92  E1        pop   hl
0F93  CD130C    call  0C13          SCR NEXT LINE
0F96  10C9      djnz  0F61
0F98  C9        ret

0F99  E5        push  hl
0F9A  D5        push  de
0F9B  1604      ld    d,04
0F9D  7E        ld    a,(hl)
0F9E  E5        push  hl
0F9F  A9        xor   c
0FA0  F5        push  af
0FA1  21CFB1    ld    hl,B1CF       Bit Masken abh. v. Mode
0FA4  A6        and   (hl)
0FA5  2001      jr    nz,0FA8
0FA7  37        scf
0FA8  CB13      rl    e
0FAA  F1        pop   af
0FAB  23        inc   hl
0FAC  A6        and   (hl)
0FAD  2001      jr    nz,0FB0
0FAF  37        scf
0FB0  CB13      rl    e
0FB2  E1        pop   hl
0FB3  CDF90B    call  0BF9          SCR NEXT BYTE
0FB6  15        dec   d
0FB7  20E4      jr    nz,0F9D
0FB9  E1        pop   hl
0FBA  73        ld    (hl),e
0FBB  EB        ex    de,hl
0FBC  13        inc   de
0FBD  E1        pop   hl
0FBE  CD130C    call  0C13          SCR NEXT LINE
0FC1  10D6      djnz  0F99
0FC3  C9        ret
```

```
************************************************ SCR HORIZONTAL
0FC4  F5        push  af
0FC5  E5        push  hl
0FC6  7A        ld    a,d
0FC7  2F        cpl
0FC8  67        ld    h,a
0FC9  7B        ld    a,e
0FCA  2F        cpl
0FCB  6F        ld    l,a
0FCC  23        inc   hl
0FCD  09        add   hl,bc
0FCE  23        inc   hl
0FCF  E3        ex    (sp),hl
0FD0  AF        xor   a
0FD1  93        sub   e
0FD2  F5        push  af
0FD3  CDA90B    call  0BA9        SCR DOT POSITION
0FD6  E5        push  hl
0FD7  78        ld    a,b
0FD8  2F        cpl
0FD9  6F        ld    l,a
0FDA  26FF      ld    h,FF
0FDC  2207B2    ld    (B207),hl
0FDF  E1        pop   hl
0FE0  F1        pop   af
0FE1  A0        and   b
0FE2  47        ld    b,a
0FE3  2845      jr    z,102A
0FE5  E3        ex    (sp),hl
0FE6  1803      jr    0FEB
0FE8  1A        ld    a,(de)
0FE9  B1        or    c
0FEA  4F        ld    c,a
0FEB  2B        dec   hl
0FEC  7C        ld    a,h
0FED  B5        or    l
0FEE  2834      jr    z,1024
0FF0  13        inc   de
0FF1  10F5      djnz  0FE8
0FF3  EB        ex    de,hl
0FF4  E1        pop   hl
0FF5  F1        pop   af
0FF6  47        ld    b,a
0FF7  CDE8BD    call  BDE8        SCR WRITE
0FFA  CDF90B    call  0BF9        SCR NEXT BYTE
0FFD  E5        push  hl
0FFE  2A07B2    ld    hl,(B207)
1001  19        add   hl,de
1002  300C      jr    nc,1010
1004  EB        ex    de,hl
1005  E1        pop   hl
1006  0EFF      ld    c,FF
1008  CDE8BD    call  BDE8        SCR WRITE
100B  CDF90B    call  0BF9        SCR NEXT BYTE
100E  18ED      jr    0FFD
```

```
1010  7B        ld    a,e
1011  B7        or    a
1012  280E      jr    z,1022
1014  AF        xor   a
1015  21CFB1    ld    hl,B1CF      Bit Masken abh. v. Mode
1018  B6        or    (hl)
1019  23        inc   hl
101A  1D        dec   e
101B  20FB      jr    nz,1018
101D  4F        ld    c,a
101E  E1        pop   hl
101F  C3E8BD    jp    BDE8         SCR WRITE

1022  E1        pop   hl
1023  C9        ret

1024  E1        pop   hl
1025  F1        pop   af
1026  47        ld    b,a
1027  C3E8BD    jp    BDE8         SCR WRITE

102A  D1        pop   de
102B  F1        pop   af
102C  47        ld    b,a
102D  18CE      jr    0FFD

*********************************************** SCR VERTICAL
102F  F5        push  af
1030  E5        push  hl
1031  7C        ld    a,h
1032  2F        cpl
1033  67        ld    h,a
1034  7D        ld    a,l
1035  2F        cpl
1036  6F        ld    l,a
1037  23        inc   hl
1038  09        add   hl,bc
1039  23        inc   hl
103A  E3        ex    (sp),hl
103B  CDA90B    call  0BA9         SCR DOT POSITION
103E  D1        pop   de
103F  F1        pop   af
1040  47        ld    b,a
1041  CDE8BD    call  BDE8         SCR WRITE
1044  CD2D0C    call  0C2D         SCR PREV LINE
1047  1B        dec   de
1048  7A        ld    a,d
1049  B3        or    e
104A  20F5      jr    nz,1041
104C  C9        ret
```

```
************************************************** Default Farben
104D  04 04 0A 13 0C 0B 14 15
1055  0D 06 1E 1F 07 12 19 04
105D  17 04 04 0A 13 0C 0B 14
1065  15 0D 06 1E 1F 07 12 19
106D  0A 07

106F  C7                rst    0
1070  C7                rst    0
1071  C7                rst    0
1072  C7                rst    0
1073  C7                rst    0
1074  C7                rst    0
1075  C7                rst    0
1076  C7                rst    0
1077  C7                rst    0
```

2.5.5 TEXT SCREEN (TXT)

Dieses Pack ist, wie der Name schon sagt, für die Verwaltung von Texten verantwortlich. Dazu gehört auch die Organisation der Windows.
Zu der Handhabung des Cursor sind einige Worte notwendig:
Die in den Cursor-Routinen verlangten oder gelieferten Koordinaten sind als logische Angaben zu verstehen, d.h. sie beziehen sich auf das laufende Fenster. Die Koordinate 1,1 ist dabei die linke obere Ecke des Fensters.
Wollen Sie, z.B. mit *TXT SET CUSOR*, den Cursor außerhalb des Fensters positionieren, wird er auomatisch auf die nächst mögliche Position innerhalb des Fensters gesetzt, falls der Cursor eingeschaltet ist oder nachfolgend ein Zeichen dargestellt werden soll.
Dadurch wird auch die laufende Position (die Sie mit *TXT GET CURSOR* zurückbekommen) geändert.
Ist der Cursor ausgeschaltet, wird die gewünschte neue Position zunächst akzeptiert, bis entweder ein Zeichen dargestellt oder der Cursor eingeschaltet wird.

Es gibt zwei Routinen, um den Cursor ein- oder auszuschalten. *TXT CUR ON/OFF* und *TXT CUR ENABLE/DISABLE* unterscheiden sich dadurch, daß *ON/OFF* der Routine *ENABLE/DISABLE* übergeordnet ist. Das heißt, daß der Cursor nach *ENABLE* nur erscheinen kann, wenn es auch mit *ON* erlaubt wurde.

Zusätzlich zu den in Kapitel 2.3 bereits vorgestellten Routinen möchten wir noch eine weitere erwähnen:
TXT OUTPUT bringt das Zeichen in **a** auf das lfd. Bildschirmfenster, bzw. führt es aus, falls es sich um ein Steuerzeichen handelte.
Beachten Sie, daß diese Routine die Indirection *TXT OUT ACTION* benutzt! Falls Sie diese 'verbogen' haben, wird auch Ihre Routine benutzt und nicht die ROM-Routine.

```
*************************************************** TXT INITIALISE
1078  CD8810      call  1088        TXT RESET
107B  AF          xor   a
107C  3295B2      ld    (B295),a
107F  210100      ld    hl,0001
1082  CD3D11      call  113D        TXT Default Params setzen
1085  C3A310      jp    10A3        Reset Params (alle Fenster)

***************************************************** TXT RESET
1088  219110      ld    hl,1091     Restore TXT Indirections
108B  CD8A0A      call  0A8A        Move (hl+3)⇒((hl+1)),cnt=(hl)
108E  C35B14      jp    145B

1091  0F          db    0F          15 Bytes
1092  CDBD        dw    BDCD        Zieladresse
1094  C36312      jp    1263        TXT DRAW/UNDRAW CURSOR

1097  C36312      jp    1263        TXT DRAW/UNDRAW CURSOR

109A  C34A13      jp    134A        TXT WRITE CHAR

109D  C3C013      jp    13C0        TXT UNWRITE

10A0  C30C14      jp    140C        TXT OUT ACTION

******************************************* Reset Params (alle Fenster)
10A3  3E08        ld    a,08
10A5  110DB2      ld    de,B20D     Start Params Fenster 0
10A8  2185B2      ld    hl,B285     lfd Cursor Pos.(Row,Col)
10AB  010F00      ld    bc,000F
10AE  EDB0        ldir
10B0  3D          dec   a
10B1  20F5        jr    nz,10A8
10B3  320CB2      ld    (B20C),a    (lfd Bildschirmfenster)
10B6  C9          ret

10B7  3A0CB2      ld    a,(B20C)    (lfd Bildschirmfenster)
10BA  4F          ld    c,a
10BB  0608        ld    b,08
10BD  78          ld    a,b
10BE  3D          dec   a
10BF  CDE810      call  10E8        TXT STR SELECT
10C2  CDD0BD      call  BDD0        TXT UNDRAW CURSOR
10C5  CDC312      call  12C3        TXT GET PAPER
10C8  3290B2      ld    (B290),a    (TXT lfd Paper)
10CB  CDBD12      call  12BD        TXT GET PEN
10CE  328FB2      ld    (B28F),a    (TXT lfd Pen)
10D1  10EA        djnz  10BD
10D3  79          ld    a,c
10D4  C9          ret
```

```
10D5  4F          ld     c,a
10D6  0608        ld     b,08
10D8  78          ld     a,b
10D9  3D          dec    a
10DA  CDE810      call   10E8          TXT STR SELECT
10DD  C5          push   bc
10DE  2A8FB2      ld     hl,(B28F)     (TXT lfd Pen)
10E1  CD3D11      call   113D          TXT Default Params setzen
10E4  C1          pop    bc
10E5  10F1        djnz   10D8
10E7  79          ld     a,c

************************************************ TXT STR SELECT
10E8  E607        and    07
10EA  210CB2      ld     hl,B20C       lfd Bildschirmfenster
10ED  BE          cp     (hl)
10EE  C8          ret    z
10EF  C5          push   bc
10F0  D5          push   de
10F1  4E          ld     c,(hl)
10F2  77          ld     (hl),a
10F3  47          ld     b,a
10F4  79          ld     a,c
10F5  CD2A11      call   112A          Adr. Fenster Params ⇨ de
10F8  CD2211      call   1122          ldir cnt=15
10FB  78          ld     a,b
10FC  CD2A11      call   112A          Adr. Fenster Params ⇨ de
10FF  EB          ex     de,hl
1100  CD2211      call   1122          ldir cnt=15
1103  79          ld     a,c
1104  D1          pop    de
1105  C1          pop    bc
1106  C9          ret

********************************************** TXT SWAP STREAMS
1107  3A0CB2      ld     a,(B20C)      (lfd Bildschirmfenster)
110A  F5          push   af
110B  79          ld     a,c
110C  CDE810      call   10E8          TXT STR SELECT
110F  78          ld     a,b
1110  320CB2      ld     (B20C),a      (lfd Bildschirmfenster)
1113  CD2A11      call   112A          Adr. Fenster Params ⇨ de
1116  D5          push   de
1117  79          ld     a,c
1118  CD2A11      call   112A          Adr. Fenster Params ⇨ de
111B  E1          pop    hl
111C  CD2211      call   1122          ldir cnt=15
111F  F1          pop    af
1120  18C6        jr     10E8          TXT STR SELECT
```

```
***************************************************** ldir cnt=15
1122  C5          push  bc
1123  010F00      ld    bc,000F
1126  EDB0        ldir
1128  C1          pop   bc
1129  C9          ret

******************************************** Adr. Fenster Params ⇒ de
112A  E607        and   07
112C  5F          ld    e,a
112D  87          add   a,a
112E  87          add   a,a
112F  87          add   a,a
1130  87          add   a,a
1131  93          sub   e
1132  C60D        add   a,0D
1134  5F          ld    e,a
1135  CEB2        adc   a,B2
1137  93          sub   e
1138  57          ld    d,a
1139  2185B2      ld    hl,B285        lfd Cursor Pos.(Row,Col)
113C  C9          ret

******************************************** TXT Default Params setzen
113D  EB          ex    de,hl
113E  3E03        ld    a,03
1140  328DB2      ld    (B28D),a       (lfd Cursor Flag)
1143  7A          ld    a,d
1144  CDAE12      call  12AE           TXT SET PAPER
1147  7B          ld    a,e
1148  CDA912      call  12A9           TXT SET PEN
114B  AF          xor   a
114C  CDA713      call  13A7           TXT SET GRAPHIC
114F  CD7A13      call  137A           TXT SET BACK
1152  210000      ld    hl,0000
1155  117F7F      ld    de,7F7F
1158  CD0C12      call  120C           TXT WIN ENABLE
115B  C35114      jp    1451           TXT VDU ENABLE

************************************************ TXT SET COLUMN
115E  3D          dec   a
115F  2189B2      ld    hl,B289        lfd Fenster links
1162  86          add   a,(hl)
1163  2A85B2      ld    hl,(B285)      (lfd Cursor Pos.(Row,Col))
1166  67          ld    h,a
1167  180E        jr    1177

*************************************************** TXT SET ROW
1169  3D          dec   a
116A  2188B2      ld    hl,B288        lfd Fenster oben
116D  86          add   a,(hl)
116E  2A85B2      ld    hl,(B285)      (lfd Cursor Pos.(Row,Col))
1171  6F          ld    l,a
1172  1803        jr    1177
```

```
********************************************* TXT SET CURSOR
1174  CD8A11      call  118A        lfd Fenst. oben,links+hl
1177  CDD0BD      call  BDD0        TXT UNDRAW CURSOR
117A  2285B2      ld    (B285),hl   (lfd Cursor Pos.(Row,Col))
117D  C3CDBD      jp    BDCD        TXT DRAW CURSOR

********************************************* TXT GET CURSOR
1180  2A85B2      ld    hl,(B285)   (lfd Cursor Pos.(Row,Col))
1183  CD9711      call  1197        lfd Fenster oben,links-hl
1186  3A8CB2      ld    a,(B28C)    (lfd Roll Count)
1189  C9          ret

******************************************* lfd Fenst. oben,links+hl
118A  3A88B2      ld    a,(B288)    (lfd Fenster oben)
118D  3D          dec   a
118E  85          add   a,l
118F  6F          ld    l,a
1190  3A89B2      ld    a,(B289)    (lfd Fenster links)
1193  3D          dec   a
1194  84          add   a,h
1195  67          ld    h,a
1196  C9          ret

******************************************* lfd Fenster oben,links-hl
1197  3A88B2      ld    a,(B288)    (lfd Fenster oben)
119A  95          sub   l
119B  2F          cpl
119C  3C          inc   a
119D  3C          inc   a
119E  6F          ld    l,a
119F  3A89B2      ld    a,(B289)    (lfd Fenster links)
11A2  94          sub   h
11A3  2F          cpl
11A4  3C          inc   a
11A5  3C          inc   a
11A6  67          ld    h,a
11A7  C9          ret

************************************************* move Cursor
11A8  CDD0BD      call  BDD0        TXT UNDRAW CURSOR
11AB  2A85B2      ld    hl,(B285)   (lfd Cursor Pos.(Row,Col))
11AE  CDDA11      call  11DA        hl innerhalb Fenstergrenzen?
11B1  2285B2      ld    (B285),hl   (lfd Cursor Pos.(Row,Col))
11B4  D8          ret   c
11B5  E5          push  hl
11B6  218CB2      ld    hl,B28C     lfd Roll Count
11B9  78          ld    a,b
11BA  87          add   a,a
11BB  3C          inc   a
11BC  86          add   a,(hl)
11BD  77          ld    (hl),a
11BE  CD5612      call  1256        TXT GET WINDOW
11C1  3A90B2      ld    a,(B290)    (TXT lfd Paper)
11C4  F5          push  af
11C5  DC3E0E      call  c,0E3E      SCR SW ROLL
```

```
11C8  F1          pop   af
11C9  D4FA0D      call  nc,0DFA        SCR HW ROLL
11CC  E1          pop   hl
11CD  C9          ret

*********************************************************** TXT VALIDATE
11CE  CD8A11      call  118A           lfd Fenst. oben,links+hl
11D1  CDDA11      call  11DA           hl innerhalb Fenstergrenzen?
11D4  F5          push  af
11D5  CD9711      call  1197           lfd Fenster oben,links-hl
11D8  F1          pop   af
11D9  C9          ret

**************************************************** hl innerhalb Fenstergrenzen?
11DA  3A8BB2      ld    a,(B28B)       (lfd Fenster rechts)
11DD  BC          cp    h
11DE  F2E611      jp    p,11E6
11E1  3A89B2      ld    a,(B289)       (lfd Fenster links)
11E4  67          ld    h,a
11E5  2C          inc   l
11E6  3A89B2      ld    a,(B289)       (lfd Fenster links)
11E9  3D          dec   a
11EA  BC          cp    h
11EB  FAF311      jp    m,11F3
11EE  3A8BB2      ld    a,(B28B)       (lfd Fenster rechts)
11F1  67          ld    h,a
11F2  2D          dec   l
11F3  3A88B2      ld    a,(B288)       (lfd Fenster oben)
11F6  3D          dec   a
11F7  BD          cp    l
11F8  F20612      jp    p,1206
11FB  3A8AB2      ld    a,(B28A)       (lfd Fenster unten)
11FE  BD          cp    l
11FF  37          scf
1200  F0          ret   p
1201  6F          ld    l,a
1202  06FF        ld    b,FF
1204  B7          or    a
1205  C9          ret

1206  3C          inc   a
1207  6F          ld    l,a
1208  0600        ld    b,00
120A  B7          or    a
120B  C9          ret

********************************************************** TXT WIN ENABLE
120C  CD570B      call  0B57           SCR CHAR LIMITS
120F  7C          ld    a,h
1210  CD4412      call  1244
1213  67          ld    h,a
1214  7A          ld    a,d
1215  CD4412      call  1244
1218  57          ld    d,a
1219  BC          cp    h
```

```
121A  3002        jr    nc,121E
121C  54          ld    d,h
121D  67          ld    h,a
121E  7D          ld    a,l
121F  CD4D12      call  124D
1222  6F          ld    l,a
1223  7B          ld    a,e
1224  CD4D12      call  124D
1227  5F          ld    e,a
1228  BD          cp    l
1229  3002        jr    nc,122D
122B  5D          ld    e,l
122C  6F          ld    l,a
122D  2288B2      ld    (B288),hl      (lfd Fenster oben)
1230  ED538AB2    ld    (B28A),de      (lfd Fenster unten)
1234  7C          ld    a,h
1235  B5          or    l
1236  2006        jr    nz,123E
1238  7A          ld    a,d
1239  A8          xor   b
123A  2002        jr    nz,123E
123C  7B          ld    a,e
123D  A9          xor   c
123E  3287B2      ld    (B287),a       (Fenst.Flag(0=ges. Bildsch.))
1241  C37711      jp    1177

1244  B7          or    a
1245  F24912      jp    p,1249
1248  AF          xor   a
1249  B8          cp    b
124A  D8          ret   c
124B  78          ld    a,b
124C  C9          ret

124D  B7          or    a
124E  F25212      jp    p,1252
1251  AF          xor   a
1252  B9          cp    c
1253  D8          ret   c
1254  79          ld    a,c
1255  C9          ret

******************************************* TXT GET WINDOW
1256  2A88B2      ld    hl,(B288)      (lfd Fenster oben)
1259  ED5B8AB2    ld    de,(B28A)      (lfd Fenster unten)
125D  3A87B2      ld    a,(B287)       (Fenst.Flag(0=ges. Bildsch.))
1260  C6FF        add   a,FF
1262  C9          ret
```

```
******************************************** TXT DRAW/UNDRAW CURSOR
1263  3A8DB2      ld    a,(B28D)      (lfd Cursor Flag)
1266  B7          or    a
1267  C0          ret   nz

******************************************** TXT PLACE/REMOVE CURSOR
1268  C5          push  bc
1269  D5          push  de
126A  E5          push  hl
126B  CDAB11      call  11AB
126E  ED4B8FB2    ld    bc,(B28F)     (TXT lfd Pen)
1272  CDDF0D      call  0DDF          SCR CHAR INVERT
1275  E1          pop   hl
1276  D1          pop   de
1277  C1          pop   bc
1278  C9          ret

**************************************************** TXT CUR ON
1279  F5          push  af
127A  3EFD        ld    a,FD
127C  CD8B12      call  128B          Cur Enable Cont'd
127F  F1          pop   af
1280  C9          ret

**************************************************** TXT CUR OFF
1281  F5          push  af
1282  3E02        ld    a,02
1284  CD9C12      call  129C          Cur Disable Cont'd
1287  F1          pop   af
1288  C9          ret

************************************************* TXT CUR ENABLE
1289  3EFE        ld    a,FE

************************************************* Cur Enable Cont'd
128B  F5          push  af
128C  CDD0BD      call  BDD0          TXT UNDRAW CURSOR
128F  F1          pop   af
1290  E5          push  hl
1291  218DB2      ld    hl,B28D       lfd Cursor Flag
1294  A6          and   (hl)
1295  77          ld    (hl),a
1296  E1          pop   hl
1297  C3CDBD      jp    BDCD          TXT DRAW CURSOR

************************************************ TXT CUR DISABLE
129A  3E01        ld    a,01

************************************************ Cur Disable Cont'd
129C  F5          push  af
129D  CDD0BD      call  BDD0          TXT UNDRAW CURSOR
12A0  F1          pop   af
12A1  E5          push  hl
12A2  218DB2      ld    hl,B28D       lfd Cursor Flag
12A5  B6          or    (hl)
```

```
12A6  77           ld    (hl),a
12A7  E1           pop   hl
12A8  C9           ret

*********************************************** TXT SET PEN
12A9  218FB2       ld    hl,B28F       TXT lfd Pen
12AC  1803         jr    12B1

********************************************** TXT SET PAPER
12AE  2190B2       ld    hl,B290       TXT lfd Paper
12B1  F5           push  af
12B2  CDD0BD       call  BDD0          TXT UNDRAW CURSOR
12B5  F1           pop   af
12B6  CD860C       call  0C86          SCR INK ENCODE
12B9  77           ld    (hl),a
12BA  C3CDBD       jp    BDCD          TXT DRAW CURSOR

*********************************************** TXT GET PEN
12BD  3A8FB2       ld    a,(B28F)      (TXT lfd Pen)
12C0  C3A00C       jp    0CA0          SCR INK DECODE

******************************************** TXT GET PAPER
12C3  3A90B2       ld    a,(B290)      (TXT lfd Paper)
12C6  C3A00C       jp    0CA0          SCR INK DECODE

*********************************************** TXT INVERSE
12C9  2A8FB2       ld    hl,(B28F)     (TXT lfd Pen)
12CC  7C           ld    a,h
12CD  65           ld    h,l
12CE  6F           ld    l,a
12CF  228FB2       ld    (B28F),hl     (TXT lfd Pen)
12D2  C9           ret

******************************************** TXT GET MATRIX
12D3  D5           push  de
12D4  5F           ld    e,a
12D5  CD2A13       call  132A          TXT GET M TABLE
12D8  3009         jr    nc,12E3
12DA  57           ld    d,a
12DB  7B           ld    a,e
12DC  92           sub   d
12DD  3F           ccf
12DE  3003         jr    nc,12E3
12E0  5F           ld    e,a
12E1  1803         jr    12E6
12E3  210038       ld    hl,3800
12E6  F5           push  af
12E7  1600         ld    d,00
12E9  EB           ex    de,hl
12EA  29           add   hl,hl
12EB  29           add   hl,hl
12EC  29           add   hl,hl
12ED  19           add   hl,de
12EE  F1           pop   af
12EF  D1           pop   de
```

```
12F0  C9                ret

****************************************************** TXT SET MATRIX
12F1  EB                ex    de,hl
12F2  CDD312            call  12D3          TXT GET MATRIX
12F5  D0                ret   nc
12F6  EB                ex    de,hl
12F7  010800            ld    bc,0008
12FA  EDB0              ldir
12FC  C9                ret

****************************************************** TXT SET M TABLE
12FD  E5                push  hl
12FE  7A                ld    a,d
12FF  B7                or    a
1300  1600              ld    d,00
1302  2019              jr    nz,131D
1304  15                dec   d
1305  D5                push  de
1306  4B                ld    c,e
1307  EB                ex    de,hl
1308  79                ld    a,c
1309  CDD312            call  12D3          TXT GET MATRIX
130C  7C                ld    a,h
130D  AA                xor   d
130E  2004              jr    nz,1314
1310  7D                ld    a,l
1311  AB                xor   e
1312  2808              jr    z,131C
1314  C5                push  bc
1315  CDF712            call  12F7
1318  C1                pop   bc
1319  0C                inc   c
131A  20EC              jr    nz,1308
131C  D1                pop   de
131D  CD2A13            call  132A          TXT GET M TABLE
1320  ED5394B2          ld    (B294),de     (1. Zeichen User Matrix)
1324  D1                pop   de
1325  ED5396B2          ld    (B296),de     (Adr. User Matrix)
1329  C9                ret

****************************************************** TXT GET M TABLE
132A  2A94B2            ld    hl,(B294)     (1. Zeichen User Matrix)
132D  7C                ld    a,h
132E  0F                rrca
132F  7D                ld    a,l
1330  2A96B2            ld    hl,(B296)     (Adr. User Matrix)
1333  C9                ret
```

```
************************************************** TXT WR CHAR
1334  47          ld    b,a
1335  3A8EB2      ld    a,(B28E)     (VDU Flag (0=disabled))
1338  B7          or    a
1339  C8          ret   z
133A  C5          push  bc
133B  CDA811      call  11A8         move Cursor
133E  24          inc   h
133F  2285B2      ld    (B285),hl    (lfd Cursor Pos.(Row,Col))
1342  25          dec   h
1343  F1          pop   af
1344  CDD3BD      call  BDD3         TXT WRITE CHAR
1347  C3CDBD      jp    BDCD         TXT DRAW CURSOR

************************************************** TXT WRITE CHAR
134A  E5          push  hl
134B  CDD312      call  12D3         TXT GET MATRIX
134E  1198B2      ld    de,B298
1351  D5          push  de
1352  CDF30E      call  0EF3         SCR UNPACK
1355  D1          pop   de
1356  E1          pop   hl
1357  CD640B      call  0B64         SCR CHAR POSITION
135A  0E08        ld    c,08
135C  C5          push  bc
135D  E5          push  hl
135E  C5          push  bc
135F  D5          push  de
1360  EB          ex    de,hl
1361  4E          ld    c,(hl)
1362  CD7613      call  1376
1365  CDF90B      call  0BF9         SCR NEXT BYTE
1368  D1          pop   de
1369  13          inc   de
136A  C1          pop   bc
136B  10F1        djnz  135E
136D  E1          pop   hl
136E  CD130C      call  0C13         SCR NEXT LINE
1371  C1          pop   bc
1372  0D          dec   c
1373  20E7        jr    nz,135C
1375  C9          ret

1376  2A91B2      ld    hl,(B291)    (lfd Background Mode)
1379  E9          jp    (hl)

************************************************** TXT SET BACK
137A  219113      ld    hl,1391
137D  B7          or    a
137E  2803        jr    z,1383
1380  219F13      ld    hl,139F
1383  2291B2      ld    (B291),hl    (lfd Background Mode)
1386  C9          ret
```

```
*********************************************** TXT GET BACK
1387  2A91B2      ld    hl,(B291)     (lfd Background Mode)
138A  116FEC      ld    de,EC6F
138D  19          add   hl,de
138E  7C          ld    a,h
138F  B5          or    l
1390  C9          ret

1391  2A8FB2      ld    hl,(B28F)     (TXT lfd Pen)
1394  79          ld    a,c
1395  2F          cpl
1396  A4          and   h
1397  47          ld    b,a
1398  79          ld    a,c
1399  A5          and   l
139A  B0          or    b
139B  0EFF        ld    c,FF
139D  1803        jr    13A2
139F  3A8FB2      ld    a,(B28F)      (TXT lfd Pen)
13A2  47          ld    b,a
13A3  EB          ex    de,hl
13A4  C36B0C      jp    0C6B          SCR PIXELS

********************************************* TXT SET GRAPHIC
13A7  3293B2      ld    (B293),a      (GRA CHAR WR Mode (0=disabl))
13AA  C9          ret

************************************************ TXT RD CHAR
13AB  E5          push  hl
13AC  D5          push  de
13AD  C5          push  bc
13AE  CDD0BD      call  BDD0          TXT UNDRAW CURSOR
13B1  2A85B2      ld    hl,(B285)     (lfd Cursor Pos.(Row,Col))
13B4  CDD6BD      call  BDD6          TXT UNWRITE
13B7  F5          push  af
13B8  CDCDBD      call  BDCD          TXT DRAW CURSOR
13BB  F1          pop   af
13BC  C1          pop   bc
13BD  D1          pop   de
13BE  E1          pop   hl
13BF  C9          ret

********************************************** TXT UNWRITE
13C0  3A8FB2      ld    a,(B28F)      (TXT lfd Pen)
13C3  1198B2      ld    de,B298
13C6  E5          push  hl
13C7  D5          push  de
13C8  CD490F      call  0F49          SCR REPACK
13CB  CDE313      call  13E3
13CE  D1          pop   de
13CF  E1          pop   hl
13D0  3001        jr    nc,13D3
13D2  C0          ret   nz
13D3  3A90B2      ld    a,(B290)      (TXT lfd Paper)
13D6  D5          push  de
```

```
13D7  CD490F      call  0F49        SCR REPACK
13DA  D1          pop   de
13DB  0608        ld    b,08
13DD  1A          ld    a,(de)
13DE  2F          cpl
13DF  12          ld    (de),a
13E0  13          inc   de
13E1  10FA        djnz  13DD
13E3  0E00        ld    c,00
13E5  79          ld    a,c
13E6  CDD312      call  12D3        TXT GET MATRIX
13E9  1198B2      ld    de,B298
13EC  0608        ld    b,08
13EE  1A          ld    a,(de)
13EF  BE          cp    (hl)
13F0  2009        jr    nz,13FB
13F2  23          inc   hl
13F3  13          inc   de
13F4  10F8        djnz  13EE
13F6  79          ld    a,c
13F7  FE20        cp    20
13F9  37          scf
13FA  C9          ret

13FB  0C          inc   c
13FC  20E7        jr    nz,13E5
13FE  AF          xor   a
13FF  C9          ret

************************************************** TXT OUTPUT
1400  F5          push  af
1401  C5          push  bc
1402  D5          push  de
1403  E5          push  hl
1404  CDD9BD      call  BDD9        TXT OUT ACTION
1407  E1          pop   hl
1408  D1          pop   de
1409  C1          pop   bc
140A  F1          pop   af
140B  C9          ret

************************************************ TXT OUT ACTION
140C  4F          ld    c,a
140D  3A93B2      ld    a,(B293)    (GRA CHAR WR Mode (0=disabl))
1410  B7          or    a
1411  79          ld    a,c
1412  C24519      jp    nz,1945     GRA WR CHAR
1415  21B8B2      ld    hl,B2B8     Zeichenzähler Control Buffer
1418  46          ld    b,(hl)
1419  78          ld    a,b
141A  FE0A        cp    0A          Control Buffer voll ?
141C  3028        jr    nc,1446     ja ⇨
141E  B7          or    a           leer ?
141F  2006        jr    nz,1427     nein ⇨
1421  79          ld    a,c
```

```
1422  FE20      cp    20          Steuerzeichen ?
1424  D23413    jp    nc,1334     nein ⇨ TXT WR CHAR
1427  04        inc   b           Zähler +1
1428  70        ld    (hl),b
1429  58        ld    e,b
142A  1600      ld    d,00
142C  19        add   hl,de
142D  71        ld    (hl),c
142E  3AB9B2    ld    a,(B2B9)    (Start Control Buffer)
1431  5F        ld    e,a
1432  21C3B2    ld    hl,B2C3     Sprungtabelle Steuerzeichen
1435  19        add   hl,de
1436  19        add   hl,de
1437  19        add   hl,de
1438  7E        ld    a,(hl)      erforderl. Anzahl
1439  B8        cp    b           Steuerparameter erreicht ?
143A  D0        ret   nc          nein ⇨
143B  23        inc   hl
143C  5E        ld    e,(hl)
143D  23        inc   hl
143E  56        ld    d,(hl)
143F  21B9B2    ld    hl,B2B9     Start Control Buffer
1442  79        ld    a,c
1443  CD1600    call  0016        call (de)
1446  AF        xor   a
1447  32B8B2    ld    (B2B8),a    (Zeichenzähler Control Buffer)
144A  C9        ret

****************************************** TXT VDU DISABLE
144B  CD9A12    call  129A        TXT CUR DISABLE
144E  AF        xor   a
144F  1805      jr    1456

****************************************** TXT VDU ENABLE
1451  CD8912    call  1289        TXT CUR ENABLE
1454  3EFF      ld    a,FF
1456  328EB2    ld    (B28E),a    (VDU Flag (0=disabled))
1459  18EB      jr    1446
145B  AF        xor   a
145C  32B8B2    ld    (B2B8),a    (Zeichenzähler Control Buffer)
145F  216B14    ld    hl,146B     Default Steuerzeichen Sprünge
1462  11C3B2    ld    de,B2C3     Sprungtabelle Steuerzeichen
1465  016000    ld    bc,0060
1468  EDB0      ldir
146A  C9        ret

*************************************** Default Steuerzeichen Sprünge
146B  00        db    00
146C  E214      dw    14E2        00 kein Effekt
146E  00        db    00
146F  3413      dw    1334        01 TXT WR CHAR
1471  00        db    00
1472  9A12      dw    129A        02 TXT CUR DISABLE
1474  00        db    00
1475  8912      dw    1289        03 TXT CUR ENABLE
```

```
1477  01     db  01
1478  CA0A   dw  0ACA   04 SCR SET MODE
147A  01     db  01
147B  4519   dw  1945   05 GRA WR CHAR
147D  00     db  00
147E  5114   dw  1451   06 TXT VDU ENABLE
1480  00     db  00
1481  D814   dw  14D8   07 Klingel
1483  00     db  00
1484  0A15   dw  150A   08 CRSR LEFT
1486  00     db  00
1487  0F15   dw  150F   09 CRSR RGHT
1489  00     db  00
148A  1415   dw  1514   0A CRSR DOWN
148C  00     db  00
148D  1915   dw  1519   0B CRSR UP
148F  00     db  00
1490  4015   dw  1540   0C TXT CLEAR WINDOW
1492  00     db  00
1493  3015   dw  1530   0D CRSR auf Zeilenanfang
1495  01     db  01
1496  AE12   dw  12AE   0E TXT SET PAPER
1498  01     db  01
1499  A912   dw  12A9   0F TXT SET PEN
149B  00     db  00
149C  4F15   dw  154F   10 Zeichen auf CRS Pos löschen
149E  00     db  00
149F  8E15   dw  158E   11 Zeile bis CRS Pos löschen
14A1  00     db  00
14A2  8415   dw  1584   12 Zeile ab CRS Pos löschen
14A4  00     db  00
14A5  6D15   dw  156D   13 Fenster bis CRS Pos lösche
14A7  00     db  00
14A8  5615   dw  1556   14 Fenster ab CRS Pos löschen
14AA  00     db  00
14AB  4B14   dw  144B   15 TXT VDU DISABLE
14AD  01     db  01
14AE  E314   dw  14E3   16 Transparentmode Ein/Aus
14B0  01     db  01
14B1  490C   dw  0C49   17 SCR ACCESS
14B3  00     db  00
14B4  C912   dw  12C9   18 TXT INVERSE
14B6  09     db  09
14B7  0415   dw  1504   19 =SYMBOL Befehl
14B9  04     db  04
14BA  F814   dw  14F8   1A Fenster definieren
14BC  00     db  00
14BD  E214   dw  14E2   1B kein Effekt
14BF  03     db  03
14C0  E814   dw  14E8   1C =INK Befehl
14C2  02     db  02
14C3  F114   dw  14F1   1D =BORDER Befehl
14C5  00     db  00
14C6  2A15   dw  152A   1E CRSR HOME
14C8  02     db  02
```

```
14C9  3815      dw    1538         1F =LOCATE Befehl

*************************************************** TXT GET CONTROLS
14CB  21C3B2    ld    hl,B2C3      Sprungtabelle Steuerzeichen
14CE  C9        ret

14CF  87        add   a,a
14D0  00        nop
14D1  00        nop
14D2  5A        ld    e,d
14D3  00        nop
14D4  00        nop
14D5  0B        dec   bc
14D6  14        inc   d
14D7  00        nop

************************************************************ 07 Klingel
14D8  DDE5      push  ix
14DA  21CF14    ld    hl,14CF
14DD  CD9F1F    call  1F9F         SOUND QUEUE
14E0  DDE1      pop   ix
14E2  C9        ret

******************************************** 16 Transparentmode Ein/Aus
14E3  0F        rrca
14E4  9F        sbc   a,a
14E5  C37A13    jp    137A         TXT SET BACK

***************************************************** 1C =INK Befehl
14E8  23        inc   hl
14E9  7E        ld    a,(hl)
14EA  23        inc   hl
14EB  46        ld    b,(hl)
14EC  23        inc   hl
14ED  4E        ld    c,(hl)
14EE  C3EC0C    jp    0CEC         SCR SET INK

************************************************** 1D =BORDER Befehl
14F1  23        inc   hl
14F2  46        ld    b,(hl)
14F3  23        inc   hl
14F4  4E        ld    c,(hl)
14F5  C3F10C    jp    0CF1         SCR SET BORDER

************************************************** 1A Fenster definieren
14F8  23        inc   hl
14F9  56        ld    d,(hl)
14FA  23        inc   hl
14FB  7E        ld    a,(hl)
14FC  23        inc   hl
14FD  5E        ld    e,(hl)
14FE  23        inc   hl
14FF  6E        ld    l,(hl)
1500  67        ld    h,a
1501  C30C12    jp    120C         TXT WIN ENABLE
```

```
*********************************************** 19 =SYMBOL Befehl
1504  23       inc   hl
1505  7E       ld    a,(hl)
1506  23       inc   hl
1507  C3F112   jp    12F1          TXT SET MATRIX

**************************************************** 08 CRSR LEFT
150A  1100FF   ld    de,FF00
150D  180D     jr    151C

**************************************************** 09 CRSR RGHT
150F  110001   ld    de,0100
1512  1808     jr    151C

**************************************************** 0A CRSR DOWN
1514  110100   ld    de,0001
1517  1803     jr    151C

****************************************************** 0B CRSR UP
1519  11FF00   ld    de,00FF
151C  D5       push  de
151D  CDA811   call  11A8          move Cursor
1520  D1       pop   de
1521  7D       ld    a,l
1522  83       add   a,e
1523  6F       ld    l,a
1524  7C       ld    a,h
1525  82       add   a,d
1526  67       ld    h,a
1527  C37A11   jp    117A

**************************************************** 1E CRSR HOME
152A  2A88B2   ld    hl,(B288)     (lfd Fenster oben)
152D  C37711   jp    1177

******************************************* 0D CRSR auf Zeilenanfang
1530  CDA811   call  11A8          move Cursor
1533  3A89B2   ld    a,(B289)      (lfd Fenster links)
1536  18EE     jr    1526

************************************************* 1F =LOCATE Befehl
1538  23       inc   hl
1539  56       ld    d,(hl)
153A  23       inc   hl
153B  5E       ld    e,(hl)
153C  EB       ex    de,hl
153D  C37411   jp    1174          TXT SET CURSOR
```

```
****************************************** TXT CLEAR WINDOW
1540  CDD0BD     call  BDD0       TXT UNDRAW CURSOR
1543  2A88B2     ld    hl,(B288)  (lfd Fenster oben)
1546  2285B2     ld    (B285),hl  (lfd Cursor Pos.(Row,Col))
1549  ED5B8AB2   ld    de,(B28A)  (lfd Fenster unten)
154D  1848       jr    1597

**************************************** 10 Zeichen auf CRS Pos löschen
154F  CDA811     call  11A8       move Cursor
1552  54         ld    d,h
1553  5D         ld    e,l
1554  1841       jr    1597

***************************************** 14 Fenster ab CRS Pos löschen
1556  CD8415     call  1584       12 Zeile ab CRS Pos löschen
1559  2A88B2     ld    hl,(B288)  (lfd Fenster oben)
155C  ED5B8AB2   ld    de,(B28A)  (lfd Fenster unten)
1560  3A85B2     ld    a,(B285)   (lfd Cursor Pos.(Row,Col))
1563  6F         ld    l,a
1564  2C         inc   l
1565  BB         cp    e
1566  3A90B2     ld    a,(B290)   (TXT lfd Paper)
1569  DCB30D     call  c,0DB3     SCR FILL BOX
156C  C9         ret

***************************************** 13 Fenster bis CRS Pos löschen
156D  CD8E15     call  158E       11 Zeile bis CRS Pos löschen
1570  2A88B2     ld    hl,(B288)  (lfd Fenster oben)
1573  3A8BB2     ld    a,(B28B)   (lfd Fenster rechts)
1576  57         ld    d,a
1577  3A85B2     ld    a,(B285)   (lfd Cursor Pos.(Row,Col))
157A  3D         dec   a
157B  5F         ld    e,a
157C  BD         cp    l
157D  3A90B2     ld    a,(B290)   (TXT lfd Paper)
1580  D4B30D     call  nc,0DB3    SCR FILL BOX
1583  C9         ret

****************************************** 12 Zeile ab CRS Pos löschen
1584  CDA811     call  11A8       move Cursor
1587  5D         ld    e,l
1588  3A8BB2     ld    a,(B28B)   (lfd Fenster rechts)
158B  57         ld    d,a
158C  1809       jr    1597

****************************************** 11 Zeile bis CRS Pos löschen
158E  CDA811     call  11A8       move Cursor
1591  EB         ex    de,hl
1592  6B         ld    l,e
1593  3A89B2     ld    a,(B289)   (lfd Fenster links)
1596  67         ld    h,a
1597  3A90B2     ld    a,(B290)   (TXT lfd Paper)
159A  CDB30D     call  0DB3       SCR FILL BOX
159D  CDCDBD     call  BDCD       TXT DRAW CURSOR
15A0  C9         ret
```

```
15A1  C7        rst   0
15A2  C7        rst   0
15A3  C7        rst   0
15A4  C7        rst   0
15A5  C7        rst   0
15A6  C7        rst   0
15A7  C7        rst   0
15A8  C7        rst   0
15A9  C7        rst   0
15AA  C7        rst   0
15AB  C7        rst   0
15AC  C7        rst   0
15AD  C7        rst   0
15AE  C7        rst   0
15AF  C7        rst   0
```

2.5.6 GRAPHICS SCREEN (GRA)

Dieses Pack dient ausschließlich der Handhabung des Grafikfensters. Zu den Koordinatenangaben, die von den verschiedenen Routinen verlangt werden, ist folgendes zu bemerken:
Die Koordinaten werden in 3(4) Stufen übersetzt. Die anwendernächste Stufe ist die Position bezüglich des von ihm gesetzten Koordinatenursprungs (*ORIGIN*). Diese wird umgerechnet in eine Position relativ zum Bildschirmursprung (unten links).
Diese beiden Stufen sind vom Mode unabhängig!
Die letzte Stufe ist die physikalische Adresse des Punktes. **Diese ist abhängig vom lfd. Modus!**
Evtl. wird oben noch eine weitere Ebene vorgesetzt, nämlich dann, wenn ein relatives Koordinatenpaar in eine absolute Position relativ zu ORIGIN berechnet werden muß.

Interessante Routinen sind: *GRA PLOT ABSOLUTE* setzt einen Punkt an der durch **de** (X-Koordinate) und **hl** (Y-Koordinate) gegebenen absoluten Position, falls diese innerhalb des Grafikfensters liegt.
Beachten Sie, daß diese Routine über die Indirection *GRA PLOT* läuft, in deren Verlauf auch die Indirection *SCR WRITE* benutzt wird!

GRA LINE ABSOLUTE zeichnet eine Linie vom laufenden Grafikcursor bis zu der durch **de** (X-Koordinate) und **hl** (Y-Koordinate) bestimmten absoluten Position, falls diese innerhalb des Grafikfensters liegt. Auch hier werden wieder Indirections benutzt, und zwar *GRA LINE* und *SCR WRITE*!

GRA WR CHAR bringt das in **a** enthaltene Zeichen auf den Bildschirm, und zwar an der laufenden Grafikcursor-Position. Diese legt die linke obere Ecke des Zeichens fest. Der Grafikcursor wird danach um die Zeichenbreite weitergesetzt. Diese Distanz ist vom Modus abhängig!

```
****************************************************** GRA INITIALISE
15B0  CDDF15        call   15DF         GRA RESET
15B3  210100        ld     hl,0001      Pen 1 , Paper 0
15B6  7C            ld     a,h
15B7  CDFD17        call   17FD         GRA SET PAPER
15BA  7D            ld     a,l
15BB  CDF617        call   17F6         GRA SET PEN
15BE  210000        ld     hl,0000      Origin auf 0,0 setzen
15C1  54            ld     d,h
15C2  5D            ld     e,l
15C3  CD0416        call   1604         GRA SET ORIGIN
15C6  110080        ld     de,8000
15C9  21FF7F        ld     hl,7FFF
15CC  E5            push   hl
15CD  D5            push   de
15CE  CD3417        call   1734         GRA WIN WIDTH
15D1  E1            pop    hl
15D2  D1            pop    de
15D3  C37917        jp     1779         GRA WIN HEIGHT

15D6  CD0A18        call   180A         GRA GET PAPER
15D9  67            ld     h,a
15DA  CD0418        call   1804         GRA GET PEN
15DD  6F            ld     l,a
15DE  C9            ret

******************************************************** GRA RESET
15DF  21E515        ld     hl,15E5      Restore GRA Indirections
15E2  C38A0A        jp     0A8A         Move (hl+3)⇒((hl+1)),cnt=(hl)

15E5  09            db     09           9 Bytes
15E6  DCBD          dw     BDDC         Zieladresse
15E8  C31618        jp     1816         GRA PLOT

15EB  C32A18        jp     182A         GRA TEST

15EE  C33C18        jp     183C         GRA LINE

**************************************************** GRA MOVE RELATIVE
15F1  CD5716        call   1657         Add lfd Koord. + rel Koord.
15F4  ED532CB3      ld     (B32C),de    (lfd X Koord.)
15F8  222EB3        ld     (B32E),hl    (lfd Y Koord.)
15FB  C9            ret

****************************************************** GRA ASK CURSOR
15FC  ED5B2CB3      ld     de,(B32C)    (lfd X Koord.)
1600  2A2EB3        ld     hl,(B32E)    (lfd Y Koord.)
1603  C9            ret

1604  ED5328B3      ld     (B328),de    (X Origin)
1608  222AB3        ld     (B32A),hl    (Y Origin)
160B  110000        ld     de,0000
160E  62            ld     h,d
160F  6B            ld     l,e
```

```
1610  18E2        jr    15F4          GRA MOVE ABSOLUTE

********************************************** GRA GET ORIGIN
1612  ED5B28B3    ld    de,(B328)     (X Origin)
1616  2A2AB3      ld    hl,(B32A)     (Y Origin)
1619  C9          ret

****************************************** phys Startposition holen
161A  CDFC15      call  15FC          GRA ASK CURSOR

**************************************** phys Zielpos holen+Cur setzen
161D  CDF415      call  15F4          GRA MOVE ABSOLUTE
1620  E5          push  hl
1621  CDEC0A      call  0AEC          SCR GET MODE
1624  2F          cpl
1625  C601        add   a,01
1627  CE02        adc   a,02
1629  2600        ld    h,00
162B  6F          ld    l,a
162C  CB7A        bit   7,d
162E  2803        jr    z,1633
1630  EB          ex    de,hl
1631  19          add   hl,de
1632  EB          ex    de,hl
1633  2F          cpl
1634  A3          and   e
1635  5F          ld    e,a
1636  7D          ld    a,l
1637  2A28B3      ld    hl,(B328)     (X Origin)
163A  19          add   hl,de
163B  0F          rrca
163C  DC7417      call  c,1774
163F  0F          rrca
1640  DC7417      call  c,1774
1643  D1          pop   de
1644  E5          push  hl
1645  7A          ld    a,d
1646  07          rlca
1647  3001        jr    nc,164A
1649  13          inc   de
164A  7B          ld    a,e
164B  E6FE        and   FE
164D  5F          ld    e,a
164E  2A2AB3      ld    hl,(B32A)     (Y Origin)
1651  19          add   hl,de
1652  CD7417      call  1774
1655  D1          pop   de
1656  C9          ret

****************************************** Add lfd Koord. + rel Koord.
1657  E5          push  hl
1658  2A2CB3      ld    hl,(B32C)     (lfd X Koord.)
165B  19          add   hl,de
165C  D1          pop   de
165D  E5          push  hl
```

```
165E  2A2EB3      ld    hl,(B32E)      (lfd Y Koord.)
1661  19          add   hl,de
1662  D1          pop   de
1663  C9          ret

1664  D5          push  de
1665  E5          push  hl
1666  2A30B3      ld    hl,(B330)      (X Koord GRA Fenster links)
1669  2B          dec   hl
166A  B7          or    a
166B  ED52        sbc   hl,de
166D  F2AC16      jp    p,16AC
1670  2A32B3      ld    hl,(B332)      (X Koord GRA Fenster rechts)
1673  B7          or    a
1674  ED52        sbc   hl,de
1676  FAAC16      jp    m,16AC
1679  D1          pop   de
167A  2A34B3      ld    hl,(B334)      (Y Koord GRA Fenster oben)
167D  B7          or    a
167E  ED52        sbc   hl,de
1680  FAAD16      jp    m,16AD
1683  2A36B3      ld    hl,(B336)      (Y Koord GRA Fenster unten)
1686  2B          dec   hl
1687  B7          or    a
1688  ED52        sbc   hl,de
168A  FA9116      jp    m,1691
168D  ED5B36B3    ld    de,(B336)      (Y Koord GRA Fenster unten)
1691  2A36B3      ld    hl,(B336)      (Y Koord GRA Fenster unten)
1694  2B          dec   hl
1695  B7          or    a
1696  ED42        sbc   hl,bc
1698  F2AD16      jp    p,16AD
169B  2A34B3      ld    hl,(B334)      (Y Koord GRA Fenster oben)
169E  B7          or    a
169F  ED42        sbc   hl,bc
16A1  F2A816      jp    p,16A8
16A4  ED4B34B3    ld    bc,(B334)      (Y Koord GRA Fenster oben)
16A8  EB          ex    de,hl
16A9  D1          pop   de
16AA  37          scf
16AB  C9          ret

16AC  E1          pop   hl
16AD  D1          pop   de
16AE  B7          or    a
16AF  C9          ret

16B0  E5          push  hl
16B1  D5          push  de
16B2  EB          ex    de,hl
16B3  2A36B3      ld    hl,(B336)      (Y Koord GRA Fenster unten)
16B6  2B          dec   hl
16B7  B7          or    a
16B8  ED52        sbc   hl,de
16BA  F2F816      jp    p,16F8
```

```
16BD  2A34B3      ld    hl,(B334)    (Y Koord GRA Fenster oben)
16C0  B7          or    a
16C1  ED52        sbc   hl,de
16C3  FAF816      jp    m,16F8
16C6  D1          pop   de
16C7  2A32B3      ld    hl,(B332)    (X Koord GRA Fenster rechts)
16CA  B7          or    a
16CB  ED52        sbc   hl,de
16CD  FAF916      jp    m,16F9
16D0  2A30B3      ld    hl,(B330)    (X Koord GRA Fenster links)
16D3  2B          dec   hl
16D4  B7          or    a
16D5  ED52        sbc   hl,de
16D7  FADE16      jp    m,16DE
16DA  ED5B30B3    ld    de,(B330)    (X Koord GRA Fenster links)
16DE  2A30B3      ld    hl,(B330)    (X Koord GRA Fenster links)
16E1  2B          dec   hl
16E2  B7          or    a
16E3  ED42        sbc   hl,bc
16E5  F2F916      jp    p,16F9
16E8  2A32B3      ld    hl,(B332)    (X Koord GRA Fenster rechts)
16EB  B7          or    a
16EC  ED42        sbc   hl,bc
16EE  F2F516      jp    p,16F5
16F1  ED4B32B3    ld    bc,(B332)    (X Koord GRA Fenster rechts)
16F5  E1          pop   hl
16F6  37          scf
16F7  C9          ret

16F8  D1          pop   de
16F9  E1          pop   hl
16FA  B7          or    a
16FB  C9          ret

16FC  CD1D16      call  161D         phys Zielpos holen+Cur setzen
16FF  E5          push  hl
1700  2A30B3      ld    hl,(B330)    (X Koord GRA Fenster links)
1703  2B          dec   hl
1704  B7          or    a
1705  ED52        sbc   hl,de
1707  F22D17      jp    p,172D
170A  2A32B3      ld    hl,(B332)    (X Koord GRA Fenster rechts)
170D  B7          or    a
170E  ED52        sbc   hl,de
1710  FA2D17      jp    m,172D
1713  E1          pop   hl
1714  D5          push  de
1715  EB          ex    de,hl
1716  2A36B3      ld    hl,(B336)    (Y Koord GRA Fenster unten)
1719  2B          dec   hl
171A  B7          or    a
171B  ED52        sbc   hl,de
171D  F23017      jp    p,1730
1720  2A34B3      ld    hl,(B334)    (Y Koord GRA Fenster oben)
1723  B7          or    a
```

```
1724  ED52        sbc   hl,de
1726  FA3017      jp    m,1730
1729  EB          ex    de,hl
172A  D1          pop   de
172B  37          scf
172C  C9          ret

172D  E1          pop   hl
172E  B7          or    a
172F  C9          ret

1730  EB          ex    de,hl
1731  D1          pop   de
1732  B7          or    a
1733  C9          ret

********************************************* GRA WIN WIDTH
1734  E5          push  hl
1735  CD6017      call  1760
1738  D1          pop   de
1739  E5          push  hl
173A  CD6017      call  1760
173D  D1          pop   de
173E  7B          ld    a,e
173F  95          sub   l
1740  7A          ld    a,d
1741  9C          sbc   a,h
1742  3801        jr    c,1745
1744  EB          ex    de,hl
1745  7B          ld    a,e
1746  E6F8        and   F8
1748  5F          ld    e,a
1749  7D          ld    a,l
174A  F607        or    07
174C  6F          ld    l,a
174D  CDEC0A      call  0AEC           SCR GET MODE
1750  3D          dec   a
1751  FC7017      call  m,1770
1754  3D          dec   a
1755  FC7017      call  m,1770
1758  ED5330B3    ld    (B330),de      (X Koord GRA Fenster links)
175C  2232B3      ld    (B332),hl      (X Koord GRA Fenster rechts)
175F  C9          ret

1760  7A          ld    a,d
1761  B7          or    a
1762  210000      ld    hl,0000
1765  F8          ret   m
1766  217F02      ld    hl,027F
1769  7B          ld    a,e
176A  95          sub   l
176B  7A          ld    a,d
176C  9C          sbc   a,h
176D  D0          ret   nc
176E  EB          ex    de,hl
```

```
176F  C9          ret

1770  CB2A        sra   d
1772  CB1B        rr    e
1774  CB2C        sra   h
1776  CB1D        rr    l
1778  C9          ret

********************************************* GRA WIN HEIGHT
1779  E5          push  hl
177A  CD9217      call  1792
177D  D1          pop   de
177E  E5          push  hl
177F  CD9217      call  1792
1782  D1          pop   de
1783  7D          ld    a,l
1784  93          sub   e
1785  7C          ld    a,h
1786  9A          sbc   a,d
1787  3801        jr    c,178A
1789  EB          ex    de,hl
178A  ED5334B3    ld    (B334),de     (Y Koord GRA Fenster oben)
178E  2236B3      ld    (B336),hl     (Y Koord GRA Fenster unten)
1791  C9          ret

1792  7A          ld    a,d
1793  B7          or    a
1794  210000      ld    hl,0000
1797  F8          ret   m
1798  CB3A        srl   d
179A  CB1B        rr    e
179C  21C700      ld    hl,00C7
179F  7B          ld    a,e
17A0  95          sub   l
17A1  7A          ld    a,d
17A2  9C          sbc   a,h
17A3  D0          ret   nc
17A4  EB          ex    de,hl
17A5  C9          ret

********************************************* GRA GET W WIDTH
17A6  ED5B30B3    ld    de,(B330)     (X Koord GRA Fenster links)
17AA  2A32B3      ld    hl,(B332)     (X Koord GRA Fenster rechts)
17AD  CDEC0A      call  0AEC          SCR GET MODE
17B0  3D          dec   a
17B1  FCB617      call  m,17B6
17B4  3D          dec   a
17B5  F0          ret   p
17B6  29          add   hl,hl
17B7  23          inc   hl
17B8  EB          ex    de,hl
17B9  29          add   hl,hl
17BA  EB          ex    de,hl
17BB  C9          ret
```

```
******************************************** GRA GET W HEIGHT
17BC  ED5B34B3      ld     de,(B334)     (Y Koord GRA Fenster oben)
17C0  2A36B3        ld     hl,(B336)     (Y Koord GRA Fenster unten)
17C3  18F1          jr     17B6

******************************************** GRA CLEAR WINDOW
17C5  CDA617        call   17A6          GRA GET W WIDTH
17C8  B7            or     a
17C9  ED52          sbc    hl,de
17CB  23            inc    hl
17CC  CD7417        call   1774
17CF  CD7417        call   1774
17D2  CB3D          srl    l
17D4  45            ld     b,l
17D5  ED5B36B3      ld     de,(B336)     (Y Koord GRA Fenster unten)
17D9  2A34B3        ld     hl,(B334)     (Y Koord GRA Fenster oben)
17DC  E5            push   hl
17DD  B7            or     a
17DE  ED52          sbc    hl,de
17E0  23            inc    hl
17E1  4D            ld     c,l
17E2  ED5B30B3      ld     de,(B330)     (X Koord GRA Fenster links)
17E6  E1            pop    hl
17E7  C5            push   bc
17E8  CDA90B        call   0BA9          SCR DOT POSITION
17EB  D1            pop    de
17EC  3A39B3        ld     a,(B339)      (GRA Paper)
17EF  4F            ld     c,a
17F0  CDB70D        call   0DB7          SCR FLOOD BOX
17F3  C30B16        jp     160B

************************************************** GRA SET PEN
17F6  CD860C        call   0C86          SCR INK ENCODE
17F9  3238B3        ld     (B338),a      (GRA Pen)
17FC  C9            ret

*********************************************** GRA SET PAPER
17FD  CD860C        call   0C86          SCR INK ENCODE
1800  3239B3        ld     (B339),a      (GRA Paper)
1803  C9            ret

************************************************ GRA GET PEN
1804  3A38B3        ld     a,(B338)      (GRA Pen)
1807  C3A00C        jp     0CA0          SCR INK DECODE

*********************************************** GRA GET PAPER
180A  3A39B3        ld     a,(B339)      (GRA Paper)
180D  C3A00C        jp     0CA0          SCR INK DECODE

******************************************** GRA PLOT RELATIVE
1810  CD5716        call   1657          Add lfd Koord. + rel Koord.
```

```
**************************************** GRA PLOT ABSOLUTE
1813  C3DCBD       jp     BDDC           GRA PLOT

*********************************************** GRA PLOT
1816  CDFC16       call   16FC
1819  D0           ret    nc
181A  CDA90B       call   0BA9           SCR DOT POSITION
181D  3A38B3       ld     a,(B338)       (GRA Pen)
1820  47           ld     b,a
1821  C3E8BD       jp     BDE8           SCR WRITE

**************************************** GRA TEST RELATIVE
1824  CD5716       call   1657           Add lfd Koord. + rel Koord.

**************************************** GRA TEST ABSOLUTE
1827  C3DFBD       jp     BDDF           GRA TEST

*********************************************** GRA TEST
182A  CDFC16       call   16FC
182D  D20A18       jp     nc,180A        GRA GET PAPER
1830  CDA90B       call   0BA9           SCR DOT POSITION
1833  C3E5BD       jp     BDE5           SCR READ

**************************************** GRA LINE RELATIVE
1836  CD5716       call   1657           Add lfd Koord. + rel Koord.

**************************************** GRA LINE ABSOLUTE
1839  C3E2BD       jp     BDE2           GRA LINE

*********************************************** GRA LINE
183C  E5           push   hl
183D  D5           push   de
183E  CD1A16       call   161A           phys Startposition holen
1841  ED5342B3     ld     (B342),de      (Rechenpuffer X Koord)
1845  2244B3       ld     (B344),hl      (Rechenpuffer Y Koord)
1848  D1           pop    de
1849  E1           pop    hl
184A  CD1D16       call   161D           phys Zielpos holen+Cur setzen
184D  E5           push   hl
184E  2A42B3       ld     hl,(B342)      (Rechenpuffer X Koord)
1851  B7           or     a
1852  ED52         sbc    hl,de
1854  44           ld     b,h
1855  4D           ld     c,l
1856  FA6918       jp     m,1869
1859  2A42B3       ld     hl,(B342)      (Rechenpuffer X Koord)
185C  EB           ex     de,hl
185D  2242B3       ld     (B342),hl      (Rechenpuffer X Koord)
1860  2A44B3       ld     hl,(B344)      (Rechenpuffer Y Koord)
1863  E3           ex     (sp),hl
1864  2244B3       ld     (B344),hl      (Rechenpuffer Y Koord)
1867  1808         jr     1871
1869  210000       ld     hl,0000
186C  B7           or     a
186D  ED42         sbc    hl,bc
```

```
186F  44         ld    b,h
1870  4D         ld    c,l
1871  D1         pop   de
1872  2A44B3     ld    hl,(B344)     (Rechenpuffer Y Koord)
1875  B7         or    a
1876  ED52       sbc   hl,de
1878  EB         ex    de,hl
1879  F28E18     jp    p,188E
187C  210000     ld    hl,0000
187F  B7         or    a
1880  ED52       sbc   hl,de
1882  54         ld    d,h
1883  5D         ld    e,l
1884  B7         or    a
1885  ED42       sbc   hl,bc
1887  210100     ld    hl,0001
188A  3027       jr    nc,18B3
188C  180A       jr    1898
188E  62         ld    h,d
188F  6B         ld    l,e
1890  B7         or    a
1891  ED42       sbc   hl,bc
1893  21FFFF     ld    hl,FFFF
1896  3009       jr    nc,18A1
1898  223AB3     ld    (B33A),hl
189B  60         ld    h,b
189C  69         ld    l,c
189D  3EFF       ld    a,FF
189F  1819       jr    18BA
18A1  E5         push  hl
18A2  2A42B3     ld    hl,(B342)     (Rechenpuffer X Koord)
18A5  09         add   hl,bc
18A6  2242B3     ld    (B342),hl     (Rechenpuffer X Koord)
18A9  2A44B3     ld    hl,(B344)     (Rechenpuffer Y Koord)
18AC  B7         or    a
18AD  ED52       sbc   hl,de
18AF  2244B3     ld    (B344),hl     (Rechenpuffer Y Koord)
18B2  E1         pop   hl
18B3  223AB3     ld    (B33A),hl
18B6  60         ld    h,b
18B7  69         ld    l,c
18B8  EB         ex    de,hl
18B9  AF         xor   a
18BA  3246B3     ld    (B346),a
18BD  13         inc   de
18BE  ED5340B3   ld    (B340),de
18C2  23         inc   hl
18C3  CD8C37     call  378C          hl/de ⇒ hl, Rest ⇒ de
18C6  223CB3     ld    (B33C),hl
18C9  ED533EB3   ld    (B33E),de
18CD  ED4B40B3   ld    bc,(B340)
18D1  50         ld    d,b
18D2  59         ld    e,c
18D3  CB3A       srl   d
18D5  CB1B       rr    e
```

```
18D7  C5          push  bc
18D8  ED4B3CB3    ld    bc,(B33C)
18DC  2A3EB3      ld    hl,(B33E)
18DF  19          add   hl,de
18E0  EB          ex    de,hl
18E1  2A40B3      ld    hl,(B340)
18E4  B7          or    a
18E5  ED52        sbc   hl,de
18E7  3007        jr    nc,18F0
18E9  19          add   hl,de
18EA  EB          ex    de,hl
18EB  B7          or    a
18EC  ED52        sbc   hl,de
18EE  EB          ex    de,hl
18EF  03          inc   bc
18F0  D5          push  de
18F1  3A46B3      ld    a,(B346)
18F4  B7          or    a
18F5  2823        jr    z,191A
18F7  2A42B3      ld    hl,(B342)         (Rechenpuffer X Koord)
18FA  54          ld    d,h
18FB  5D          ld    e,l
18FC  09          add   hl,bc
18FD  2242B3      ld    (B342),hl         (Rechenpuffer X Koord)
1900  44          ld    b,h
1901  4D          ld    c,l
1902  0B          dec   bc
1903  2A44B3      ld    hl,(B344)         (Rechenpuffer Y Koord)
1906  E5          push  hl
1907  CDB016      call  16B0
190A  3A38B3      ld    a,(B338)          (GRA Pen)
190D  DCC40F      call  c,0FC4            SCR HORIZONTAL
1910  D1          pop   de
1911  2A3AB3      ld    hl,(B33A)
1914  19          add   hl,de
1915  2244B3      ld    (B344),hl         (Rechenpuffer Y Koord)
1918  1823        jr    193D
191A  2A44B3      ld    hl,(B344)         (Rechenpuffer Y Koord)
191D  54          ld    d,h
191E  5D          ld    e,l
191F  09          add   hl,bc
1920  2244B3      ld    (B344),hl         (Rechenpuffer Y Koord)
1923  44          ld    b,h
1924  4D          ld    c,l
1925  0B          dec   bc
1926  EB          ex    de,hl
1927  ED5B42B3    ld    de,(B342)         (Rechenpuffer X Koord)
192B  D5          push  de
192C  CD6416      call  1664
192F  3A38B3      ld    a,(B338)          (GRA Pen)
1932  DC2F10      call  c,102F            SCR VERTICAL
1935  D1          pop   de
1936  2A3AB3      ld    hl,(B33A)
1939  19          add   hl,de
193A  2242B3      ld    (B342),hl         (Rechenpuffer X Koord)
```

```
193D  D1           pop   de
193E  C1           pop   bc
193F  0B           dec   bc
1940  78           ld    a,b
1941  B1           or    c
1942  2093         jr    nz,18D7
1944  C9           ret

************************************************ GRA WR CHAR
1945  DDE5         push  ix
1947  CDD312       call  12D3        TXT GET MATRIX
194A  113AB3       ld    de,B33A
194D  D5           push  de
194E  DDE1         pop   ix
1950  010800       ld    bc,0008
1953  EDB0         ldir
1955  CD1A16       call  161A        phys Startposition holen
1958  CDFF16       call  16FF
195B  304C         jr    nc,19A9
195D  E5           push  hl
195E  D5           push  de
195F  010700       ld    bc,0007
1962  EB           ex    de,hl
1963  09           add   hl,bc
1964  EB           ex    de,hl
1965  B7           or    a
1966  ED42         sbc   hl,bc
1968  CDFF16       call  16FF
196B  D1           pop   de
196C  E1           pop   hl
196D  303A         jr    nc,19A9
196F  CDA90B       call  0BA9        SCR DOT POSITION
1972  1608         ld    d,08
1974  E5           push  hl
1975  1E08         ld    e,08
1977  CDCF19       call  19CF
197A  CB09         rrc   c
197C  DCF90B       call  c,0BF9      SCR NEXT BYTE
197F  DDCB0006     rlc   (ix+00)
1986  E1           pop   hl
1987  CD130C       call  0C13        SCR NEXT LINE
198A  DD23         inc   ix
198C  15           dec   d
198D  20E5         jr    nz,1974
198F  DDE1         pop   ix
1991  CDFC15       call  15FC        GRA ASK CURSOR
1994  EB           ex    de,hl
1995  CDEC0A       call  0AEC        SCR GET MODE
1998  010800       ld    bc,0008
199B  FE01         cp    01
199D  2804         jr    z,19A3
199F  3003         jr    nc,19A4
19A1  09           add   hl,bc
19A2  09           add   hl,bc
19A3  09           add   hl,bc
```

```
19A4  09          add   hl,bc
19A5  EB          ex    de,hl
19A6  C3F415      jp    15F4       GRA MOVE ABSOLUTE

19A9  0E08        ld    c,08
19AB  D5          push  de
19AC  0608        ld    b,08
19AE  CDFF16      call  16FF
19B1  300C        jr    nc,19BF
19B3  E5          push  hl
19B4  D5          push  de
19B5  C5          push  bc
19B6  CDA90B      call  0BA9       SCR DOT POSITION
19B9  CDCF19      call  19CF
19BC  C1          pop   bc
19BD  D1          pop   de
19BE  E1          pop   hl
19BF  DDCB0006    rlc   (ix+00)
19C6  D1          pop   de
19C7  2B          dec   hl
19C8  DD23        inc   ix
19CA  0D          dec   c
19CB  20DE        jr    nz,19AB
19CD  18C0        jr    198F
19CF  DDCB007E    bit   7,(ix+00)
19D8  3A39B3      ld    a,(B339)   (GRA Paper)
19DB  47          ld    b,a
19DC  C3E8BD      jp    BDE8       SCR WRITE

19DF  C7          rst   0
```

2.5.7 KEYBOARD MANAGER (KM)

Die Überwachung der Tastatur und die Umsetzung in brauchbare Zeichencodes obliegt diesem Pack.
Zur zyklischen Abfrage der Tasten bedient es sich des *EVENT*-Mechanismus.

Aus den zur Verfügung stehenden Routinen haben wir folgende Auswahl getroffen:

KM WAIT CHAR holt ein Zeichen aus dem Eingabepuffer, bzw. Expansion-String oder Put Back Buffer. Falls kein Zeichen verfügbar ist, kehrt die Routine nicht zurück. Es wird auf jeden Fall gewartet.
a enthält im Erfolgsfalle das eingegebene Zeichen.

KM READ CHAR übergibt ebenfalls ein Zeichen in **a**, falls eines vorhanden war, wartet jedoch nicht auf den Erfolg.
Falls nach Rückkehr aus der Routine **carry** gesetzt ist, gab es kein Zeichen zu holen.

Die Routinen *KM WAIT KEY* und *KM READ KEY* arbeiten entsprechend, jedoch wird hier nur der Eingabepuffer auf ein Zeichen untersucht.
Expansion-String und Put Back Buffer bleiben unberücksichtigt.

Mit *KM SET REPEAT* können Sie bestimmen, welche Tasten mit der Wiederholfunktion versehen werden sollen.
In **a** ist dabei die Tastennummer zu hinterlegen. **b** muß &FF enthalten, wenn die Taste repetieren soll, und 00, wenn die Wiederholfunktion für die betreffende Taste aufgehoben werden soll.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  KM INITIALISE
19E0  21021E     ld     hl,1E02
19E3  CD6D1C     call   1C6D          KM SET DELAY
19E6  AF         xor    a
19E7  320BB5     ld     (B50B),a
19EA  67         ld     h,a
19EB  6F         ld     l,a
19EC  22E7B4     ld     (B4E7),hl     (Shift Lock State)
19EF  213CB4     ld     hl,B43C
19F2  11B0FF     ld     de,FFB0
19F5  2247B5     ld     (B547),hl     (Adr. der Repeat Tabelle)
19F8  19         add    hl,de
19F9  2245B5     ld     (B545),hl     (Adr. Key CTRL Table)
19FC  19         add    hl,de
19FD  2243B5     ld     (B543),hl     (Adr. Key SHIFT Table)
1A00  19         add    hl,de
1A01  2241B5     ld     (B541),hl     (Adr. Key Translation Table)
1A04  EB         ex     de,hl
1A05  21691D     ld     hl,1D69       Key Translation Table
1A08  01FA00     ld     bc,00FA
1A0B  EDB0       ldir
1A0D  060A       ld     b,0A
1A0F  21EBB4     ld     hl,B4EB       Key State Map
1A12  3600       ld     (hl),00
1A14  23         inc    hl
1A15  10FB       djnz   1A12
1A17  060A       ld     b,0A
1A19  36FF       ld     (hl),FF
1A1B  23         inc    hl
1A1C  10FB       djnz   1A19

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  KM RESET
1A1E  CDED1C     call   1CED
1A21  CD751A     call   1A75
1A24  1146B4     ld     de,B446
1A27  219800     ld     hl,0098
1A2A  CD811A     call   1A81          KM EXP BUFFER CONT'D
1A2D  21361A     ld     hl,1A36       Restore KM Indirection
1A30  CD8A0A     call   0A8A          Move (hl+3)⇒((hl+1)),cnt=(hl)
1A33  C3821C     jp     1C82          KM DISARM BREAK

1A36  03         db     03            3 Bytes
1A37  EEBD       dw     BDEE          Zieladresse
1A39  C32F1C     jp     1C2F          KM TEST BREAK

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  KM WAIT CHAR
1A3C  CD421A     call   1A42          KM READ CHAR
1A3F  30FB       jr     nc,1A3C       KM WAIT CHAR
1A41  C9         ret
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * KM READ CHAR
1A42  E5         push  hl
1A43  21E0B4     ld    hl,B4E0         Put Back Buffer
1A46  7E         ld    a,(hl)          Zeichen holen
1A47  36FF       ld    (hl),FF         Puffer löschen
1A49  BE         cp    (hl)            war ein Zeichen da ?
1A4A  3827       jr    c,1A73          ja ⇨
1A4C  2ADEB4     ld    hl,(B4DE)       (Exp. String Pointer)
1A4F  7C         ld    a,h
1A50  B7         or    a               Exp. String vorhanden ?
1A51  2011       jr    nz,1A64         ja ⇨
1A53  CD5C1B     call  1B5C            KM READ KEY
1A56  301B       jr    nc,1A73         kein Zeichen ⇨
1A58  FE80       cp    80              Zeichen < 128 ?
1A5A  3817       jr    c,1A73          ja ⇨
1A5C  FEA0       cp    A0
1A5E  3F         ccf
1A5F  3812       jr    c,1A73
1A61  67         ld    h,a
1A62  2E00       ld    l,00
1A64  D5         push  de
1A65  CD2E1B     call  1B2E            KM GET EXPAND
1A68  3802       jr    c,1A6C
1A6A  2600       ld    h,00
1A6C  2C         inc   l
1A6D  22DEB4     ld    (B4DE),hl       (Exp. String Pointer)
1A70  D1         pop   de
1A71  30E0       jr    nc,1A53
1A73  E1         pop   hl
1A74  C9         ret

1A75  3EFF       ld    a,FF
1A77  32E0B4     ld    (B4E0),a        (Put Back Buffer)
1A7A  C9         ret

1A7B  CD811A     call  1A81            KM EXP BUFFER CONT'D
1A7E  3F         ccf
1A7F  FB         ei
1A80  C9         ret

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * KM EXP BUFFER CONT'D
1A81  F3         di
1A82  7D         ld    a,l
1A83  D631       sub   31
1A85  7C         ld    a,h
1A86  DE00       sbc   a,00
1A88  D8         ret   c
1A89  19         add   hl,de
1A8A  22E3B4     ld    (B4E3),hl       (Pointer Ende Exp Buffer)
1A8D  EB         ex    de,hl
1A8E  22E1B4     ld    (B4E1),hl       (Pointer Start Exp Buffer)
1A91  01300A     ld    bc,0A30         ASCII
1A94  3601       ld    (hl),01         0
1A96  23         inc   hl              bis
1A97  71         ld    (hl),c          9
```

```
1A98  23          inc   hl          nach
1A99  0C          inc   c           Expansion
1A9A  10F8        djnz  1A94        Buffer
1A9C  EB          ex    de,hl       Restore
1A9D  21B31A      ld    hl,1AB3     Default Exp String
1AA0  0E0A        ld    c,0A
1AA2  EDB0        ldir
1AA4  EB          ex    de,hl
1AA5  0613        ld    b,13
1AA7  AF          xor   a
1AA8  77          ld    (hl),a
1AA9  23          inc   hl
1AAA  10FC        djnz  1AA8
1AAC  22E5B4      ld    (B4E5),hl   (Pointer freier Exp Buffer)
1AAF  32DFB4      ld    (B4DF),a
1AB2  C9          ret

********************************************** Default Exp String
1AB3  01 2E 01 0D 05 52 55 4E       .....RUN
1ABB  22 0D                         ".

********************************************** KM SET EXPAND

1ABD  F3          di
1ABE  78          ld    a,b
1ABF  CD3E1B      call  1B3E        Adr. Exp String ⇨ de
1AC2  301F        jr    nc,1AE3     Token ungültig ⇨
1AC4  C5          push  bc
1AC5  D5          push  de
1AC6  E5          push  hl
1AC7  CDE51A      call  1AE5        Exp Buffer aufräumen
1ACA  3F          ccf
1ACB  E1          pop   hl
1ACC  D1          pop   de
1ACD  C1          pop   bc
1ACE  3013        jr    nc,1AE3
1AD0  1B          dec   de
1AD1  79          ld    a,c
1AD2  0C          inc   c
1AD3  12          ld    (de),a
1AD4  13          inc   de
1AD5  E7          rst   4
1AD6  23          inc   hl
1AD7  0D          dec   c
1AD8  20F9        jr    nz,1AD3
1ADA  21DFB4      ld    hl,B4DF
1ADD  78          ld    a,b
1ADE  AE          xor   (hl)
1ADF  2001        jr    nz,1AE2
1AE1  77          ld    (hl),a
1AE2  37          scf
1AE3  FB          ei
1AE4  C9          ret
```

```
********************************************** Exp Buffer aufräumen
1AE5  0600       ld    b,00
1AE7  60         ld    h,b
1AE8  6F         ld    l,a
1AE9  79         ld    a,c
1AEA  95         sub   l
1AEB  C8         ret   z
1AEC  300F       jr    nc,1AFD
1AEE  7D         ld    a,l
1AEF  69         ld    l,c
1AF0  4F         ld    c,a
1AF1  19         add   hl,de
1AF2  EB         ex    de,hl
1AF3  09         add   hl,bc
1AF4  CD221B     call  1B22         Platz f. neuen Exp String?
1AF7  2823       jr    z,1B1C
1AF9  EDB0       ldir
1AFB  181F       jr    1B1C
1AFD  4F         ld    c,a
1AFE  19         add   hl,de
1AFF  E5         push  hl
1B00  2AE5B4     ld    hl,(B4E5)    (Pointer freier Exp Buffer)
1B03  09         add   hl,bc
1B04  EB         ex    de,hl
1B05  2AE3B4     ld    hl,(B4E3)    (Pointer Ende Exp Buffer)
1B08  7D         ld    a,l
1B09  93         sub   e
1B0A  7C         ld    a,h
1B0B  9A         sbc   a,d
1B0C  E1         pop   hl
1B0D  D8         ret   c
1B0E  CD221B     call  1B22         Platz f. neuen Exp String?
1B11  2AE5B4     ld    hl,(B4E5)    (Pointer freier Exp Buffer)
1B14  2806       jr    z,1B1C
1B16  D5         push  de
1B17  1B         dec   de
1B18  2B         dec   hl
1B19  EDB8       lddr
1B1B  D1         pop   de
1B1C  ED53E5B4   ld    (B4E5),de    (Pointer freier Exp Buffer)
1B20  B7         or    a
1B21  C9         ret

******************************************** Platz f. neuen Exp String?
1B22  3AE5B4     ld    a,(B4E5)     (Pointer freier Exp Buffer)
1B25  95         sub   l
1B26  4F         ld    c,a
1B27  3AE6B4     ld    a,(B4E6)
1B2A  9C         sbc   a,h
1B2B  47         ld    b,a
1B2C  B1         or    c
1B2D  C9         ret
```

```
********************************************** KM GET EXPAND
1B2E  CD3E1B       call   1B3E          Adr. Exp String nach de
1B31  D0           ret    nc
1B32  BD           cp     l
1B33  C8           ret    z
1B34  3F           ccf
1B35  D0           ret    nc
1B36  E5           push   hl
1B37  2600         ld     h,00
1B39  19           add    hl,de
1B3A  7E           ld     a,(hl)
1B3B  E1           pop    hl
1B3C  37           scf
1B3D  C9           ret

********************************************** Adr. Exp String nach de
1B3E  E67F         and    7F            Token im gültigen
1B40  FE20         cp     20            Bereich?
1B42  D0           ret    nc            nein ⇨
1B43  E5           push   hl
1B44  2AE1B4       ld     hl,(B4E1)     (Pointer Start Exp Buffer)
1B47  110000       ld     de,0000
1B4A  3C           inc    a
1B4B  19           add    hl,de         hl um Länge des
1B4C  5E           ld     e,(hl)        Expansion String
1B4D  23           inc    hl            weitersetzen
1B4E  3D           dec    a
1B4F  20FA         jr     nz,1B4B
1B51  7B           ld     a,e
1B52  EB           ex     de,hl
1B53  E1           pop    hl
1B54  37           scf
1B55  C9           ret

********************************************** KM WAIT KEY
1B56  CD5C1B       call   1B5C          KM READ KEY
1B59  30FB         jr     nc,1B56       KM WAIT KEY
1B5B  C9           ret

********************************************** KM READ KEY
1B5C  E5           push   hl
1B5D  C5           push   bc
1B5E  CD151D       call   1D15
1B61  303A         jr     nc,1B9D
1B63  79           ld     a,c
1B64  FEEF         cp     EF
1B66  2834         jr     z,1B9C
1B68  E60F         and    0F
1B6A  87           add    a,a
1B6B  87           add    a,a
1B6C  87           add    a,a
1B6D  3D           dec    a
1B6E  3C           inc    a
1B6F  CB08         rrc    b
1B71  30FB         jr     nc,1B6E
```

```
1B73  CDA01B     call  1BA0
1B76  21E8B4     ld    hl,B4E8       Caps Lock State
1B79  CB7E       bit   7,(hl)
1B7B  280A       jr    z,1B87
1B7D  FE61       cp    61
1B7F  3806       jr    c,1B87
1B81  FE7B       cp    7B
1B83  3002       jr    nc,1B87
1B85  C6E0       add   a,E0
1B87  FEFF       cp    FF
1B89  28D3       jr    z,1B5E
1B8B  FEFE       cp    FE
1B8D  21E7B4     ld    hl,B4E7       Shift Lock State
1B90  2805       jr    z,1B97
1B92  FEFD       cp    FD            caps lock ?
1B94  23         inc   hl
1B95  2005       jr    nz,1B9C       nein ⇒
1B97  7E         ld    a,(hl)
1B98  2F         cpl                 toggle caps lock
1B99  77         ld    (hl),a
1B9A  18C2       jr    1B5E
1B9C  37         scf
1B9D  C1         pop   bc
1B9E  E1         pop   hl
1B9F  C9         ret

1BA0  CB11       rl    c
1BA2  DA481D     jp    c,1D48        KM GET CONTROL
1BA5  47         ld    b,a
1BA6  3AE7B4     ld    a,(B4E7)      (Shift Lock State)
1BA9  B1         or    c
1BAA  E640       and   40
1BAC  78         ld    a,b
1BAD  C2431D     jp    nz,1D43       KM GET SHIFT
1BB0  C33E1D     jp    1D3E          KM GET TRANSLATE

*********************************************** KM GET STATE
1BB3  2AE7B4     ld    hl,(B4E7)     (Shift Lock State)
1BB6  C9         ret

********************************************* Update Key State Map
1BB7  11FFB4     ld    de,B4FF       Multihit Kontr. zu B4F5
1BBA  21F5B4     ld    hl,B4F5       währ. Scan gedr. Keys
1BBD  CD4608     call  0846          Scan Keyboard
1BC0  3A01B5     ld    a,(B501)
1BC3  E6A0       and   A0            SHIFT/CTRL isolieren
1BC5  4F         ld    c,a
1BC6  21EDB4     ld    hl,B4ED       Key 16...23
1BC9  B6         or    (hl)
1BCA  77         ld    (hl),a
1BCB  21FFB4     ld    hl,B4FF       Multihit Kontr. zu B4F5
1BCE  11EBB4     ld    de,B4EB       Key State Map
1BD1  0600       ld    b,00
1BD3  1A         ld    a,(de)
1BD4  AE         xor   (hl)
```

```
1BD5  A6          and   (hl)
1BD6  C4481C      call  nz,1C48
1BD9  7E          ld    a,(hl)
1BDA  12          ld    (de),a
1BDB  23          inc   hl
1BDC  13          inc   de
1BDD  0C          inc   c
1BDE  79          ld    a,c
1BDF  E60F        and   0F
1BE1  FE0A        cp    0A
1BE3  20EE        jr    nz,1BD3
1BE5  79          ld    a,c
1BE6  E6A0        and   A0
1BE8  CB71        bit   6,c
1BEA  4F          ld    c,a
1BEB  C4EEBD      call  nz,BDEE      KM TEST BREAK
1BEE  78          ld    a,b
1BEF  B7          or    a
1BF0  C0          ret   nz
1BF1  2109B5      ld    hl,B509
1BF4  35          dec   (hl)
1BF5  C0          ret   nz
1BF6  2A0AB5      ld    hl,(B50A)
1BF9  EB          ex    de,hl
1BFA  42          ld    b,d
1BFB  1600        ld    d,00
1BFD  21EBB4      ld    hl,B4EB      Key State Map
1C00  19          add   hl,de
1C01  7E          ld    a,(hl)
1C02  2A47B5      ld    hl,(B547)    (Adr. der Repeat Tabelle)
1C05  19          add   hl,de
1C06  A6          and   (hl)
1C07  A0          and   b
1C08  C8          ret   z
1C09  2109B5      ld    hl,B509
1C0C  34          inc   (hl)
1C0D  3A40B5      ld    a,(B540)
1C10  B7          or    a
1C11  C0          ret   nz
1C12  79          ld    a,c
1C13  B3          or    e
1C14  4F          ld    c,a
1C15  3AE9B4      ld    a,(B4E9)     (KM Delay)
1C18  3209B5      ld    (B509),a
1C1B  CDFE1C      call  1CFE
1C1E  79          ld    a,c
1C1F  E60F        and   0F
1C21  6F          ld    l,a
1C22  60          ld    h,b
1C23  220AB5      ld    (B50A),hl
1C26  FE08        cp    08
1C28  C0          ret   nz
1C29  CB60        bit   4,b
1C2B  C0          ret   nz
1C2C  CBF1        set   6,c
```

```
1C2E  C9            ret

************************************************ KM TEST BREAK
1C2F  21F3B4        ld     hl,B4F3
1C32  CB56          bit    2,(hl)
1C34  C8            ret    z
1C35  79            ld     a,c
1C36  EEA0          xor    A0
1C38  2056          jr     nz,1C90        KM BREAK EVENT
1C3A  C5            push   bc
1C3B  23            inc    hl
1C3C  060A          ld     b,0A
1C3E  8E            adc    a,(hl)
1C3F  2B            dec    hl
1C40  10FC          djnz   1C3E
1C42  C1            pop    bc
1C43  FEA4          cp     A4
1C45  2049          jr     nz,1C90        KM BREAK EVENT
1C47  C7            rst    0
1C48  E5            push   hl
1C49  D5            push   de
1C4A  5F            ld     e,a
1C4B  2F            cpl
1C4C  3C            inc    a
1C4D  A3            and    e
1C4E  47            ld     b,a
1C4F  3AEAB4        ld     a,(B4EA)
1C52  CD181C        call   1C18
1C55  78            ld     a,b
1C56  AB            xor    e
1C57  20F1          jr     nz,1C4A
1C59  D1            pop    de
1C5A  E1            pop    hl
1C5B  C9            ret

************************************************ KM GET JOYSTICK
1C5C  3AF1B4        ld     a,(B4F1)       (Joystick 1)
1C5F  E67F          and    7F
1C61  6F            ld     l,a
1C62  3AF4B4        ld     a,(B4F4)       (Joystick 0)
1C65  E67F          and    7F
1C67  67            ld     h,a
1C68  C9            ret

************************************************ KM GET DELAY
1C69  2AE9B4        ld     hl,(B4E9)      (KM Delay)
1C6C  C9            ret

************************************************ KM SET DELAY
1C6D  22E9B4        ld     (B4E9),hl      (KM Delay)
1C70  C9            ret
```

```
************************************************ KM ARM BREAK
1C71  CD821C        call   1C82           KM DISARM BREAK
1C74  210DB5        ld     hl,B50D        Break Event Block
1C77  0640          ld     b,40
1C79  CDD201        call   01D2           KL INIT EVENT
1C7C  3EFF          ld     a,FF
1C7E  320CB5        ld     (B50C),a
1C81  C9            ret

************************************************ KM DISARM BREAK
1C82  C5            push   bc
1C83  D5            push   de
1C84  210CB5        ld     hl,B50C
1C87  3600          ld     (hl),00
1C89  23            inc    hl
1C8A  CD8502        call   0285           KL DEL SYNCHRONOUS
1C8D  D1            pop    de
1C8E  C1            pop    bc
1C8F  C9            ret

************************************************ KM BREAK EVENT
1C90  210CB5        ld     hl,B50C
1C93  7E            ld     a,(hl)
1C94  3600          ld     (hl),00
1C96  BE            cp     (hl)
1C97  C8            ret    z
1C98  C5            push   bc
1C99  D5            push   de
1C9A  23            inc    hl
1C9B  CDE201        call   01E2           KL EVENT
1C9E  0EEF          ld     c,EF
1CA0  CDFE1C        call   1CFE
1CA3  D1            pop    de
1CA4  C1            pop    bc
1CA5  C9            ret

1CA6  2A47B5        ld     hl,(B547)      (Adr. der Repeat Tabelle)
1CA9  181D          jr     1CC8           Z entspr. Key Bit setzen
1CAB  FE50          cp     50             Key > 80 ?
1CAD  D0            ret    nc             ja ⇒ ungültig
1CAE  2A47B5        ld     hl,(B547)      (Adr. der Repeat Tabelle)
1CB1  CDCD1C        call   1CCD           der Key# entspr. Bit holen
1CB4  4F            ld     c,a
1CB5  2F            cpl
1CB6  A6            and    (hl)
1CB7  77            ld     (hl),a
1CB8  79            ld     a,c
1CB9  A0            and    b              (b=$ff/00)
1CBA  B6            or     (hl)
1CBB  77            ld     (hl),a
1CBC  C9            ret
```

```
****************************************************** KM TEST KEY
1CBD F5          push  af
1CBE 3AEDB4      ld    a,(B4ED)      (Key 16...23)
1CC1 E6A0        and   A0            SHIFT/CTRL isolieren
1CC3 4F          ld    c,a
1CC4 F1          pop   af
1CC5 21EBB4      ld    hl,B4EB       Key State Map
1CC8 CDCD1C      call  1CCD          der Key# entspr. Bit holen
1CCB A6          and   (hl)          Key Bit maskieren
1CCC C9          ret

****************************************** der Key# entspr. Bit holen
1CCD D5          push  de
1CCE F5          push  af
1CCF E6F8        and   F8            Key#
1CD1 0F          rrca                /8
1CD2 0F          rrca
1CD3 0F          rrca
1CD4 5F          ld    e,a
1CD5 1600        ld    d,00
1CD7 19          add   hl,de         Key Map adressieren
1CD8 F1          pop   af
1CD9 E5          push  hl
1CDA 21E51C      ld    hl,1CE5       Bit Masken
1CDD E607        and   07            dem Key
1CDF 5F          ld    e,a           entsprechendes
1CE0 19          add   hl,de         Bit
1CE1 7E          ld    a,(hl)        laden
1CE2 E1          pop   hl
1CE3 D1          pop   de
1CE4 C9          ret

******************************************************* Bit Masken
1CE5 01 02 04 08 10 20 40 80

1CED F3          di
1CEE 213CB5      ld    hl,B53C
1CF1 3615        ld    (hl),15
1CF3 23          inc   hl
1CF4 AF          xor   a
1CF5 77          ld    (hl),a
1CF6 23          inc   hl
1CF7 3601        ld    (hl),01
1CF9 23          inc   hl
1CFA 77          ld    (hl),a
1CFB 23          inc   hl
1CFC 77          ld    (hl),a
1CFD C9          ret

1CFE 213CB5      ld    hl,B53C
1D01 B7          or    a
1D02 35          dec   (hl)
1D03 280E        jr    z,1D13
1D05 CD2C1D      call  1D2C
1D08 71          ld    (hl),c
```

```
1D09  23            inc   hl
1D0A  70            ld    (hl),b
1D0B  2140B5        ld    hl,B540
1D0E  34            inc   (hl)
1D0F  213EB5        ld    hl,B53E
1D12  37            scf
1D13  34            inc   (hl)
1D14  C9            ret

1D15  213EB5        ld    hl,B53E
1D18  B7            or    a
1D19  35            dec   (hl)
1D1A  280E          jr    z,1D2A
1D1C  CD2C1D        call  1D2C
1D1F  4E            ld    c,(hl)
1D20  23            inc   hl
1D21  46            ld    b,(hl)
1D22  2140B5        ld    hl,B540
1D25  35            dec   (hl)
1D26  213CB5        ld    hl,B53C
1D29  37            scf
1D2A  34            inc   (hl)
1D2B  C9            ret

1D2C  23            inc   hl
1D2D  34            inc   (hl)
1D2E  7E            ld    a,(hl)
1D2F  FE14          cp    14
1D31  2002          jr    nz,1D35
1D33  AF            xor   a
1D34  77            ld    (hl),a
1D35  87            add   a,a
1D36  CE14          adc   a,14
1D38  6F            ld    l,a
1D39  CEB5          adc   a,B5
1D3B  95            sub   l
1D3C  67            ld    h,a
1D3D  C9            ret

******************************************** KM GET TRANSLATE
1D3E  2A41B5        ld    hl,(B541)     (Adr. Key Transl. Table)
1D41  1808          jr    1D4B          Get Key Table

********************************************** KM GET SHIFT
1D43  2A43B5        ld    hl,(B543)     (Adr. Key SHIFT Table)
1D46  1803          jr    1D4B          Get Key Table

********************************************* KM GET CONTROL
1D48  2A45B5        ld    hl,(B545)     (Adr. Key CTRL Table)

*********************************************** Get Key Table
1D4B  85            add   a,l
1D4C  6F            ld    l,a
1D4D  8C            adc   a,h
1D4E  95            sub   l
```

```
1D4F  67                ld     h,a
1D50  7E                ld     a,(hl)
1D51  C9                ret

*************************************** KM SET TRANSLATE
1D52  2A41B5            ld     hl,(B541)      (Adr. Key Transl. Table)
1D55  1808              jr     1D5F           Set Key Table

******************************************* KM SET SHIFT
1D57  2A43B5            ld     hl,(B543)      (Adr. Key SHIFT Table)
1D5A  1803              jr     1D5F           Set Key Table

***************************************** KM SET CONTROL
1D5C  2A45B5            ld     hl,(B545)      (Adr. Key CTRL Table)

********************************************* Set Key Table
1D5F  FE50              cp     50
1D61  D0                ret    nc
1D62  85                add    a,l
1D63  6F                ld     l,a
1D64  8C                adc    a,h
1D65  95                sub    l
1D66  67                ld     h,a
1D67  70                ld     (hl),b
1D68  C9                ret

*************************************** Key Translation Table
1D69  F0 F3 F1 89 86 83 8B 8A
1D71  F2 E0 87 88 85 81 82 80
1D79  10 5B 0D 5D 84 FF 5C FF
1D81  5E 2D 40 70 3B 3A 2F 2E
1D89  30 39 6F 69 6C 6B 6D 2C
1D91  38 37 75 79 68 6A 6E 20
1D99  36 35 72 74 67 66 62 76
1DA1  34 33 65 77 73 64 63 78
1DA9  31 32 FC 71 09 61 FD 7A
1DB1  0B 0A 08 09 58 5A FF 7F

******************************************* Key SHIFT Table
1DB9  F4 F7 F5 89 86 83 8B 8A
1DC1  F6 E0 87 88 85 81 82 80
1DC9  10 7B 0D 7D 84 FF 60 FF
1DD1  A3 3D 7C 50 2B 2A 3F 3E
1DD9  5F 29 4F 49 4C 4B 4D 3C
1DE1  28 27 55 59 48 4A 4E 20
1DE9  26 25 52 54 47 46 42 56
1DF1  24 23 45 57 53 44 43 58
1DF9  21 22 FC 51 09 41 FD 5A
1E01  0B 0A 08 09 58 5A FF 7F

******************************************** Key CTRL Table
1E09  F8 FB F9 89 86 83 8C 8A
1E11  FA E0 87 88 85 81 82 80
1E19  10 1B 0D 1D 84 FF 1C FF
1E21  1E FF 00 10 FF FF FF FF
```

```
1E29  1F FF 0F 09 0C 0B 0D FF
1E31  FF FF 15 19 08 0A 0E FF
1E39  FF FF 12 14 07 06 02 16
1E41  FF FF 05 17 13 04 03 18
1E49  FF 7E FC 11 E1 01 FE 1A
1E51  FF FF FF FF FF FF FF 7F
1E59  07 03 4B FF FF FF FF FF
1E61  AB 8F

1E63  C7                rst    0
1E64  C7                rst    0
1E65  C7                rst    0
1E66  C7                rst    0
1E67  C7                rst    0
```

2.5.8 SOUND MANAGER (SOUND)

Über dieses Pack, obgleich es ziemlich umfangreich ist, gibt es nicht viel zu berichten. Die eigentliche Tonerzeugung nimmt nur wenig Raum ein. Der größte Teil ist der Verwaltung der verschiedenen Warteschlangen gewidmet. Dazu zählt auch die Realisierung der *TONE ENVELOPE*, die der PSG nicht von sich aus beherrscht.

Der engagierte Musikfan wird auch eher direkt den PSG programmieren wollen, da die Routinen des SOUND allzu speziell auf die zugehörigen BASIC-Befehle zugeschnitten sind. Zum Abspielen von Melodien, selbst dreistimmig, reicht BASIC allemal, auch bei einem flotten Tempo.

Interessant ist für den Maschinenprogrammierer allenfalls die Verwirklichung eines guten (sprich: abwechslungsreichen) Schlagzeugs, was in BASIC wegen der Verschachtelung relativ kurzer, aber komplexer Klänge nur unvollkommen möglich ist.

```
*********************************************** SOUND RESET
1E68  AF          xor   a
1E69  F3          di
1E6A  3252B5      ld    (B552),a      (lfd SOUND Aktivität)
1E6D  3251B5      ld    (B551),a      (alte SOUND Akt. (nach HOLD))
1E70  2155B5      ld    hl,B555       Sound Event Block
1E73  11031F      ld    de,1F03       Sound Event
1E76  0681        ld    b,81
1E78  CDD201      call  01D2          KL INIT EVENT
1E7B  3E3F        ld    a,3F
1E7D  3219B6      ld    (B619),a
1E80  215CB5      ld    hl,B55C       SOUND Params Kanal A
1E83  013D00      ld    bc,003D
1E86  110801      ld    de,0108
1E89  AF          xor   a
1E8A  77          ld    (hl),a
1E8B  23          inc   hl
1E8C  72          ld    (hl),d
1E8D  23          inc   hl
1E8E  73          ld    (hl),e
1E8F  09          add   hl,bc
1E90  3C          inc   a
1E91  EB          ex    de,hl
1E92  29          add   hl,hl
1E93  EB          ex    de,hl
1E94  FE03        cp    03
1E96  20F2        jr    nz,1E8A
1E98  0E07        ld    c,07
1E9A  DDE5        push  ix
1E9C  E5          push  hl
1E9D  211DB5      ld    hl,B51D
1EA0  41          ld    b,c
1EA1  113F00      ld    de,003F
1EA4  19          add   hl,de
1EA5  CB38        srl   b
1EA7  30F8        jr    nc,1EA1
1EA9  C5          push  bc
1EAA  E5          push  hl
1EAB  DDE1        pop   ix
1EAD  EB          ex    de,hl
1EAE  CD7F22      call  227F
1EB1  13          inc   de
1EB2  13          inc   de
1EB3  13          inc   de
1EB4  6B          ld    l,e
1EB5  62          ld    h,d
1EB6  13          inc   de
1EB7  013B00      ld    bc,003B
1EBA  3600        ld    (hl),00
1EBC  EDB0        ldir
1EBE  DD361C04    ld    (ix+1C),04
1EC2  C1          pop   bc
1EC3  EB          ex    de,hl
1EC4  04          inc   b
```

```
1EC5  10DE          djnz   1EA5
1EC7  E1            pop    hl
1EC8  DDE1          pop    ix
1ECA  C9            ret

*********************************************** SOUND HOLD
1ECB  2152B5        ld     hl,B552        lfd SOUND Aktivität
1ECE  F3            di
1ECF  7E            ld     a,(hl)
1ED0  3600          ld     (hl),00
1ED2  FB            ei
1ED3  B7            or     a              Kanäle aktiv ?
1ED4  C8            ret    z              nein ⇨
1ED5  2B            dec    hl
1ED6  77            ld     (hl),a
1ED7  2E03          ld     l,03           Lautstärke
1ED9  0E00          ld     c,00           aller Kanäle
1EDB  3E07          ld     a,07           auf 0
1EDD  85            add    a,l
1EDE  CD2608        call   0826           MC SOUND REGISTER
1EE1  2D            dec    l
1EE2  20F7          jr     nz,1EDB
1EE4  37            scf
1EE5  C9            ret

********************************************* SOUND CONTINUE
1EE6  3A51B5        ld     a,(B551)       (alte SOUND Akt. (nach HOLD))
1EE9  B7            or     a              Kanal aktiv ?
1EEA  C8            ret    z              nein ⇨
1EEB  DD211DB5      ld     ix,B51D
1EEF  113F00        ld     de,003F
1EF2  DD19          add    ix,de
1EF4  CB3F          srl    a              bei allen
1EF6  F5            push   af             Kanälen
1EF7  DD7E0F        ld     a,(ix+0F)      wieder alte
1EFA  DC7622        call   c,2276         Lautstärke setzen
1EFD  F1            pop    af
1EFE  20F2          jr     nz,1EF2
1F00  C31E20        jp     201E

************************************************ Sound Event
1F03  DDE5          push   ix
1F05  2150B5        ld     hl,B550
1F08  E5            push   hl
1F09  AF            xor    a
1F0A  77            ld     (hl),a
1F0B  23            inc    hl
1F0C  46            ld     b,(hl)
1F0D  C5            push   bc
1F0E  23            inc    hl             irgendein
1F0F  B6            or     (hl)           Kanal aktiv ?
1F10  2822          jr     z,1F34         nein ⇨
1F12  DD211DB5      ld     ix,B51D
1F16  013F00        ld     bc,003F
1F19  DD09          add    ix,bc
```

```
1F1B  CB3F        srl   a            Kanal aktiv ?
1F1D  30FA        jr    nc,1F19      nein ⮚ nächster
1F1F  F5          push  af
1F20  DD7E04      ld    a,(ix+04)
1F23  1F          rra
1F24  DCC222      call  c,22C2
1F27  DD7E07      ld    a,(ix+07)
1F2A  1F          rra
1F2B  DCB621      call  c,21B6
1F2E  DCA820      call  c,20A8
1F31  F1          pop   af
1F32  20E2        jr    nz,1F16
1F34  C1          pop   bc
1F35  E1          pop   hl
1F36  7E          ld    a,(hl)
1F37  B7          or    a
1F38  2820        jr    z,1F5A
1F3A  4F          ld    c,a
1F3B  23          inc   hl
1F3C  7E          ld    a,(hl)
1F3D  70          ld    (hl),b
1F3E  A8          xor   b
1F3F  47          ld    b,a
1F40  23          inc   hl
1F41  B6          or    (hl)
1F42  77          ld    (hl),a
1F43  78          ld    a,b
1F44  2F          cpl
1F45  A1          and   c
1F46  2812        jr    z,1F5A
1F48  DD211DB5    ld    ix,B51D
1F4C  113F00      ld    de,003F
1F4F  DD19        add   ix,de
1F51  CB3F        srl   a
1F53  F5          push  af
1F54  DC7F22      call  c,227F
1F57  F1          pop   af
1F58  20F5        jr    nz,1F4F
1F5A  AF          xor   a
1F5B  3254B5      ld    (B554),a
1F5E  DDE1        pop   ix
1F60  C9          ret

******************************************************* Scan Sound Queues
1F61  2152B5      ld    hl,B552      lfd SOUND Aktivität
1F64  7E          ld    a,(hl)
1F65  B7          or    a
1F66  C8          ret   z
1F67  23          inc   hl
1F68  35          dec   (hl)
1F69  C0          ret   nz
1F6A  34          inc   (hl)
1F6B  23          inc   hl
1F6C  7E          ld    a,(hl)
1F6D  B7          or    a
```

```
1F6E  C0          ret   nz
1F6F  2B          dec   hl
1F70  3603        ld    (hl),03
1F72  2B          dec   hl
1F73  46          ld    b,(hl)
1F74  2122B5      ld    hl,B522
1F77  113F00      ld    de,003F
1F7A  AF          xor   a
1F7B  19          add   hl,de
1F7C  CB38        srl   b
1F7E  30FB        jr    nc,1F7B
1F80  35          dec   (hl)
1F81  2005        jr    nz,1F88
1F83  2B          dec   hl
1F84  CB06        rlc   (hl)
1F86  8A          adc   a,d
1F87  23          inc   hl
1F88  23          inc   hl
1F89  35          dec   (hl)
1F8A  2005        jr    nz,1F91
1F8C  23          inc   hl
1F8D  CB06        rlc   (hl)
1F8F  8A          adc   a,d
1F90  2B          dec   hl
1F91  2B          dec   hl
1F92  04          inc   b
1F93  10E6        djnz  1F7B
1F95  B7          or    a
1F96  C8          ret   z
1F97  2154B5      ld    hl,B554
1F9A  77          ld    (hl),a
1F9B  23          inc   hl
1F9C  C3E201      jp    01E2          KL EVENT

************************************************ SOUND QUEUE
1F9F  CDE61E      call  1EE6          SOUND CONTINUE
1FA2  7E          ld    a,(hl)
1FA3  E607        and   07
1FA5  37          scf
1FA6  C8          ret   z
1FA7  4F          ld    c,a
1FA8  B6          or    (hl)
1FA9  FC9A1E      call  m,1E9A
1FAC  41          ld    b,c
1FAD  DD211DB5    ld    ix,B51D
1FB1  113F00      ld    de,003F
1FB4  AF          xor   a
1FB5  DD19        add   ix,de
1FB7  CB38        srl   b
1FB9  30FA        jr    nc,1FB5
1FBB  DD721E      ld    (ix+1E),d
1FBE  DDBE1C      cp    (ix+1C)
1FC1  3F          ccf
1FC2  9F          sbc   a,a
1FC3  04          inc   b
```

```
1FC4  10EF        djnz  1FB5
1FC6  B7          or    a
1FC7  C0          ret   nz
1FC8  41          ld    b,c
1FC9  7E          ld    a,(hl)
1FCA  1F          rra
1FCB  1F          rra
1FCC  1F          rra
1FCD  B0          or    b
1FCE  E60F        and   0F
1FD0  4F          ld    c,a
1FD1  23          inc   hl
1FD2  DD211DB5    ld    ix,B51D
1FD6  113F00      ld    de,003F
1FD9  DD19        add   ix,de
1FDB  CB38        srl   b
1FDD  30FA        jr    nc,1FD9
1FDF  E5          push  hl
1FE0  C5          push  bc
1FE1  DD7E1B      ld    a,(ix+1B)
1FE4  DD341B      inc   (ix+1B)
1FE7  DD351C      dec   (ix+1C)
1FEA  EB          ex    de,hl
1FEB  CD3A20      call  203A
1FEE  E5          push  hl
1FEF  EB          ex    de,hl
1FF0  DD7E01      ld    a,(ix+01)
1FF3  2F          cpl
1FF4  A1          and   c
1FF5  12          ld    (de),a
1FF6  13          inc   de
1FF7  7E          ld    a,(hl)
1FF8  23          inc   hl
1FF9  87          add   a
1FFA  87          add   a
1FFB  87          add   a
1FFC  87          add   a
1FFD  47          ld    b,a
1FFE  7E          ld    a,(hl)
1FFF  23          inc   hl
2000  E60F        and   0F
2002  B0          or    b
2003  12          ld    (de),a
2004  13          inc   de
2005  010600      ld    bc,0006
2008  EDB0        ldir
200A  E1          pop   hl
200B  F3          di
200C  DD7E1A      ld    a,(ix+1A)
200F  DD341A      inc   (ix+1A)
2012  DDB603      or    (ix+03)
2015  FB          ei
2016  CCBD20      call  z,20BD
2019  C1          pop   bc
201A  E1          pop   hl
```

```
201B  04          inc   b
201C  10B8        djnz  1FD6
201E  E5          push  hl
201F  2151B5      ld    hl,B551        alte SOUND Akt. (nach HOLD)
2022  7E          ld    a,(hl)
2023  B7          or    a
2024  2811        jr    z,2037
2026  3600        ld    (hl),00
2028  F3          di
2029  23          inc   hl
202A  46          ld    b,(hl)
202B  B0          or    b
202C  77          ld    (hl),a
202D  78          ld    a,b
202E  B7          or    a
202F  2005        jr    nz,2036
2031  23          inc   hl
2032  3603        ld    (hl),03
2034  23          inc   hl
2035  77          ld    (hl),a
2036  FB          ei
2037  E1          pop   hl
2038  37          scf
2039  C9          ret

203A  E603        and   03
203C  87          add   a,a
203D  87          add   a,a
203E  87          add   a,a
203F  C61F        add   a,1F
2041  DDE5        push  ix
2043  E1          pop   hl
2044  85          add   a,l
2045  6F          ld    l,a
2046  8C          adc   a,h
2047  95          sub   l
2048  67          ld    h,a
2049  C9          ret

********************************************** SOUND RELEASE
204A  6F          ld    l,a
204B  CDE61E      call  1EE6           SOUND CONTINUE
204E  7D          ld    a,l
204F  E607        and   07
2051  C8          ret   z
2052  DD211DB5    ld    ix,B51D
2056  113F00      ld    de,003F
2059  DD19        add   ix,de
205B  CB3F        srl   a
205D  30FA        jr    nc,2059
205F  F5          push  af
2060  DDCB035E    bit   3,(ix+03)
2068  20EC        jr    nz,2056
206A  18B2        jr    201E
```

```
****************************************************** SOUND CHECK
206C  E607       and   07
206E  C8         ret   z
206F  2120B5     ld    hl,B520
2072  113F00     ld    de,003F
2075  19         add   hl,de
2076  1F         rra
2077  30FC       jr    nc,2075
2079  F3         di
207A  7E         ld    a,(hl)
207B  87         add   a,a
207C  87         add   a,a
207D  87         add   a,a
207E  111900     ld    de,0019
2081  19         add   hl,de
2082  B6         or    (hl)
2083  23         inc   hl
2084  23         inc   hl
2085  3600       ld    (hl),00
2087  FB         ei
2088  C9         ret

**************************************************** SOUND ARM EVENT
2089  E607       and   07
208B  C8         ret   z
208C  EB         ex    de,hl
208D  2139B5     ld    hl,B539
2090  013F00     ld    bc,003F
2093  09         add   hl,bc
2094  1F         rra
2095  30FC       jr    nc,2093
2097  AF         xor   a
2098  F3         di
2099  BE         cp    (hl)
209A  23         inc   hl
209B  73         ld    (hl),e
209C  23         inc   hl
209D  2003       jr    nz,20A2
209F  72         ld    (hl),d
20A0  FB         ei
20A1  C9         ret

20A2  77         ld    (hl),a
20A3  FB         ei
20A4  EB         ex    de,hl
20A5  C3E201     jp    01E2          KL EVENT

20A8  DD7E1A     ld    a,(ix+1A)
20AB  B7         or    a
20AC  CA7F22     jp    z,227F
20AF  DD7E01     ld    a,(ix+01)
20B2  2150B5     ld    hl,B550
20B5  B6         or    (hl)
20B6  77         ld    (hl),a
20B7  DD7E19     ld    a,(ix+19)
```

```
20BA  CD3A20     call  203A
20BD  7E         ld    a,(hl)
20BE  B7         or    a
20BF  280C       jr    z,20CD
20C1  CB5F       bit   3,a
20C3  2053       jr    nz,2118
20C5  E5         push  hl
20C6  3600       ld    (hl),00
20C8  CD1F21     call  211F
20CB  E1         pop   hl
20CC  D0         ret   nc
20CD  DD360310   ld    (ix+03),10
20D1  23         inc   hl
20D2  7E         ld    a,(hl)
20D3  E6F0       and   F0
20D5  F5         push  af
20D6  AE         xor   (hl)
20D7  5F         ld    e,a
20D8  23         inc   hl
20D9  4E         ld    c,(hl)
20DA  23         inc   hl
20DB  56         ld    d,(hl)
20DC  23         inc   hl
20DD  B2         or    d
20DE  B1         or    c
20DF  2808       jr    z,20E9
20E1  E5         push  hl
20E2  CDAB22     call  22AB
20E5  DD5601     ld    d,(ix+01)
20E8  E1         pop   hl
20E9  4E         ld    c,(hl)
20EA  23         inc   hl
20EB  5E         ld    e,(hl)
20EC  23         inc   hl
20ED  7E         ld    a,(hl)
20EE  23         inc   hl
20EF  66         ld    h,(hl)
20F0  6F         ld    l,a
20F1  F1         pop   af
20F2  CD7521     call  2175
20F5  2151B5     ld    hl,B551           alte SOUND Akt. (nach HOLD)
20F8  DD7E01     ld    a,(ix+01)
20FB  B6         or    (hl)
20FC  77         ld    (hl),a
20FD  DD3419     inc   (ix+19)
2100  DD351A     dec   (ix+1A)
2103  DD341C     inc   (ix+1C)
2106  F3         di
2107  DD7E1E     ld    a,(ix+1E)
210A  DD361E00   ld    (ix+1E),00
210E  FB         ei
210F  B7         or    a
2110  C8         ret   z
2111  67         ld    h,a
2112  DD6E1D     ld    l,(ix+1D)
```

```
2115  C3E201     jp    01E2         KL EVENT

2118  CB9E       res   3,(hl)
211A  DD360308   ld    (ix+03),08
211E  C9         ret

211F  DDE5       push  ix
2121  47         ld    b,a
2122  DD4E01     ld    c,(ix+01)
2125  DD215CB5   ld    ix,B55C      SOUND Params Kanal A
2129  CB47       bit   0,a
212B  200C       jr    nz,2139
212D  DD219BB5   ld    ix,B59B      SOUND Params Kanal B
2131  CB4F       bit   1,a
2133  2004       jr    nz,2139
2135  DD21DAB5   ld    ix,B5DA      SOUND Params Kanal C
2139  F3         di
213A  DD7E03     ld    a,(ix+03)
213D  A1         and   c
213E  282D       jr    z,216D
2140  78         ld    a,b
2141  DDBE01     cp    (ix+01)
2144  281A       jr    z,2160
2146  DDE5       push  ix
2148  DD21DAB5   ld    ix,B5DA      SOUND Params Kanal C
214C  CB57       bit   2,a
214E  2004       jr    nz,2154
2150  DD219BB5   ld    ix,B59B      SOUND Params Kanal B
2154  DD7E03     ld    a,(ix+03)
2157  A1         and   c
2158  2812       jr    z,216C
215A  FB         ei
215B  CDB720     call  20B7
215E  DDE1       pop   ix
2160  DD360300   ld    (ix+03),00
2164  FB         ei
2165  CDB720     call  20B7
2168  DDE1       pop   ix
216A  37         scf
216B  C9         ret

216C  E1         pop   hl
216D  DDE1       pop   ix
216F  DD7003     ld    (ix+03),b
2172  FB         ei
2173  B7         or    a
2174  C9         ret

2175  CBFB       set   7,e
2177  DD730F     ld    (ix+0F),e
217A  5F         ld    e,a
217B  7D         ld    a,l
217C  B4         or    h
217D  2001       jr    nz,2180
217F  2B         dec   hl
```

```
2180  DD7508     ld    (ix+08),l
2183  DD7409     ld    (ix+09),h
2186  79         ld    a,c
2187  B7         or    a
2188  2808       jr    z,2192
218A  3E06       ld    a,06        Rauschgenerator laden
218C  CD2608     call  0826        MC SOUND REGISTER
218F  DD7E02     ld    a,(ix+02)
2192  B2         or    d
2193  CD8B22     call  228B
2196  7B         ld    a,e
2197  B7         or    a
2198  280A       jr    z,21A4
219A  210AB6     ld    hl,B60A     Lautstärke Hüllkurven
219D  1600       ld    d,00
219F  19         add   hl,de
21A0  7E         ld    a,(hl)
21A1  B7         or    a
21A2  2003       jr    nz,21A7
21A4  21B221     ld    hl,21B2
21A7  DD750A     ld    (ix+0A),l
21AA  DD740B     ld    (ix+0B),h
21AD  CD6522     call  2265
21B0  180D       jr    21BF
21B2  010100     ld    bc,0001
21B5  C8         ret   z
21B6  DD6E0D     ld    l,(ix+0D)
21B9  DD660E     ld    h,(ix+0E)
21BC  DD5E10     ld    e,(ix+10)
21BF  7B         ld    a,e
21C0  FEFF       cp    FF
21C2  2876       jr    z,223A
21C4  87         add   a,a
21C5  7E         ld    a,(hl)
21C6  23         inc   hl
21C7  384A       jr    c,2213
21C9  280D       jr    z,21D8
21CB  1D         dec   e
21CC  B7         or    a
21CD  2006       jr    nz,21D5
21CF  DDB60F     or    (ix+0F)
21D2  F2DD21     jp    p,21DD
21D5  DD860F     add   a,(ix+0F)
21D8  E60F       and   0F
21DA  CD7322     call  2273        Lautstärke setzen
21DD  4E         ld    c,(hl)
21DE  DD7E09     ld    a,(ix+09)
21E1  47         ld    b,a
21E2  87         add   a,a
21E3  381B       jr    c,2200
21E5  AF         xor   a
21E6  91         sub   c
21E7  DD8608     add   a,(ix+08)
21EA  380C       jr    c,21F8
21EC  05         dec   b
```

```
21ED  F2F521      jp    p,21F5
21F0  DD4E08      ld    c,(ix+08)
21F3  AF          xor   a
21F4  47          ld    b,a
21F5  DD7009      ld    (ix+09),b
21F8  DD7708      ld    (ix+08),a
21FB  B0          or    b
21FC  2002        jr    nz,2200
21FE  1EFF        ld    e,FF
2200  7B          ld    a,e
2201  B7          or    a
2202  CC4622      call  z,2246
2205  DD7310      ld    (ix+10),e
2208  F3          di
2209  DD7106      ld    (ix+06),c
220C  DD360780    ld    (ix+07),80
2210  FB          ei
2211  B7          or    a
2212  C9          ret

2213  57          ld    d,a
2214  4B          ld    c,e
2215  3E0D        ld    a,0D          Hüllkurve
2217  CD2608      call  0826          MC SOUND REGISTER
221A  4A          ld    c,d
221B  3E0B        ld    a,0B          Hüllkurvenlänge Lo
221D  CD2608      call  0826          MC SOUND REGISTER
2220  4E          ld    c,(hl)
2221  3E0C        ld    a,0C          Hüllkurvenlänge Hi
2223  CD2608      call  0826          MC SOUND REGISTER
2226  3E10        ld    a,10
2228  CD7322      call  2273          Lautstärke setzen
222B  CD4622      call  2246
222E  7B          ld    a,e
222F  3C          inc   a
2230  208D        jr    nz,21BF
2232  21B221      ld    hl,21B2
2235  CD6522      call  2265
2238  1885        jr    21BF
223A  AF          xor   a
223B  DD7703      ld    (ix+03),a
223E  DD7707      ld    (ix+07),a
2241  DD7704      ld    (ix+04),a
2244  37          scf
2245  C9          ret

2246  DD350C      dec   (ix+0C)
2249  201E        jr    nz,2269
224B  DD7E09      ld    a,(ix+09)
224E  87          add   a,a
224F  21B221      ld    hl,21B2
2252  3011        jr    nc,2265
2254  DD3408      inc   (ix+08)
2257  2006        jr    nz,225F
2259  DD3409      inc   (ix+09)
```

```
225C  1EFF       ld    e,FF
225E  C8         ret   z
225F  DD6E0A     ld    l,(ix+0A)
2262  DD660B     ld    h,(ix+0B)
2265  7E         ld    a,(hl)
2266  DD770C     ld    (ix+0C),a
2269  23         inc   hl
226A  5E         ld    e,(hl)
226B  23         inc   hl
226C  DD750D     ld    (ix+0D),l
226F  DD740E     ld    (ix+0E),h
2272  C9         ret

********************************************* Lautstärke setzen
2273  DD770F     ld    (ix+0F),a
2276  4F         ld    c,a
2277  DD7E00     ld    a,(ix+00)
227A  C608       add   a,08        Lautstärke
227C  C32608     jp    0826        MC SOUND REGISTER

227F  DD7E01     ld    a,(ix+01)
2282  2F         cpl
2283  2152B5     ld    hl,B552     lfd SOUND Aktivität
2286  F3         di
2287  A6         and   (hl)
2288  77         ld    (hl),a
2289  FB         ei
228A  AF         xor   a
228B  47         ld    b,a
228C  DD7E01     ld    a,(ix+01)
228F  DDB602     or    (ix+02)
2292  2119B6     ld    hl,B619
2295  F3         di
2296  B6         or    (hl)
2297  A8         xor   b
2298  BE         cp    (hl)
2299  77         ld    (hl),a
229A  FB         ei
229B  2003       jr    nz,22A0
229D  78         ld    a,b
229E  B7         or    a
229F  C0         ret   nz
22A0  AF         xor   a
22A1  CD7622     call  2276
22A4  F3         di
22A5  4E         ld    c,(hl)
22A6  3E07       ld    a,07        Kanal-Steuerregister
22A8  C32608     jp    0826        MC SOUND REGISTER

22AB  CD2423     call  2324
22AE  7B         ld    a,e
22AF  CD4E23     call  234E        SOUND T ADDRESS
22B2  D0         ret   nc
22B3  7E         ld    a,(hl)
22B4  E67F       and   7F
```

```
22B6  C8          ret   z
22B7  DD7511      ld    (ix+11),l
22BA  DD7412      ld    (ix+12),h
22BD  CD1323      call  2313
22C0  1809        jr    22CB
22C2  DD6E14      ld    l,(ix+14)
22C5  DD6615      ld    h,(ix+15)
22C8  DD5E18      ld    e,(ix+18)
22CB  4E          ld    c,(hl)
22CC  23          inc   hl
22CD  7B          ld    a,e
22CE  D6F0        sub   F0
22D0  3804        jr    c,22D6
22D2  1E00        ld    e,00
22D4  180E        jr    22E4
22D6  1D          dec   e
22D7  79          ld    a,c
22D8  87          add   a,a
22D9  9F          sbc   a,a
22DA  57          ld    d,a
22DB  DD7E16      ld    a,(ix+16)
22DE  81          add   a,c
22DF  4F          ld    c,a
22E0  DD7E17      ld    a,(ix+17)
22E3  8A          adc   a,d
22E4  57          ld    d,a
22E5  CD2423      call  2324
22E8  4E          ld    c,(hl)
22E9  7B          ld    a,e
22EA  B7          or    a
22EB  2019        jr    nz,2306
22ED  DD7E13      ld    a,(ix+13)
22F0  3D          dec   a
22F1  2010        jr    nz,2303
22F3  DD6E11      ld    l,(ix+11)
22F6  DD6612      ld    h,(ix+12)
22F9  7E          ld    a,(hl)
22FA  C680        add   a,80
22FC  3805        jr    c,2303
22FE  DD360400    ld    (ix+04),00
2302  C9          ret

2303  CD1323      call  2313
2306  DD7318      ld    (ix+18),e
2309  F3          di
230A  DD7105      ld    (ix+05),c
230D  DD360480    ld    (ix+04),80
2311  FB          ei
2312  C9          ret

2313  DD7713      ld    (ix+13),a
2316  23          inc   hl
2317  5E          ld    e,(hl)
2318  23          inc   hl
2319  DD7514      ld    (ix+14),l
```

```
231C  DD7415        ld     (ix+15),h
231F  7B            ld     a,e
2320  B7            or     a
2321  C0            ret    nz
2322  1C            inc    e
2323  C9            ret

2324  DD7E00        ld     a,(ix+00)
2327  87            add    a,a
2328  F5            push   af
2329  DD7116        ld     (ix+16),c      Tonhöhe Lo
232C  CD2608        call   0826           MC SOUND REGISTER
232F  F1            pop    af
2330  3C            inc    a
2331  4A            ld     c,d
2332  DD7117        ld     (ix+17),c      Tonhöhe Hi
2335  C32608        jp     0826           MC SOUND REGISTER

****************************************** SOUND AMPL ENVELOPE
2338  110AB6        ld     de,B60A        Lautstärke Hüllkurven
233B  1803          jr     2340           Hüllhurve kopieren

****************************************** SOUND TONE ENVELOPE
233D  11FAB6        ld     de,B6FA        Ton Hüllkurven

********************************************** Hüllhurve kopieren
2340  EB            ex     de,hl
2341  CD5123        call   2351           Hüllkurve Adresse holen
2344  EB            ex     de,hl
2345  D0            ret    nc
2346  EDB0          ldir
2348  C9            ret

********************************************* SOUND A ADDRESS
2349  210AB6        ld     hl,B60A        Lautstärke Hüllkurven
234C  1803          jr     2351           Hüllkurve Adresse holen

********************************************* SOUND T ADDRESS
234E  21FAB6        ld     hl,B6FA        Ton Hüllkurven

******************************************** Hüllkurve Adresse holen
2351  B7            or     a
2352  C8            ret    z
2353  FE10          cp     10
2355  D0            ret    nc
2356  011000        ld     bc,0010
2359  87            add    a,a
235A  87            add    a,a
235B  87            add    a,a
235C  87            add    a,a
235D  85            add    a,l
235E  6F            ld     l,a
235F  8C            adc    a,h
2360  95            sub    l
2361  67            ld     h,a
```

```
2362  37      scf
2363  C9      ret

2364  C7      rst   0
2365  C7      rst   0
2366  C7      rst   0
2367  C7      rst   0
2368  C7      rst   0
2369  C7      rst   0
236A  C7      rst   0
236B  C7      rst   0
236C  C7      rst   0
236D  C7      rst   0
236E  C7      rst   0
236F  C7      rst   0
```

2.5.9 CASSETTE MANAGER (CAS)

Die Aufgaben dieses Packs brauchen wir Ihnen sicher nicht näher auseinanderzusetzen.
Die Anwendung verschiedener Routinen dürfte für den Assemblerprogrammierer kaum von besonderem Interesse sein, da professionelle Programme allgemein nicht gerne im Zusammenhang mit Kassettenbetrieb gesehen werden. Die Floppy gibt in dieser Richtung schon wesentlich mehr her.
Dennoch hier einige Basisroutinen, die anwendbar sind:

CAS IN OPEN eröffnet ein Eingabefile. Dazu müssen in **b** die Länge des Filenamens, in **hl** die Anfangsadresse des Filenamens und in **de** die Startadresse eines 2K großen RAM-Bereiches, der als Eingabepuffer benutzt wird, übergeben werden.
Nach der Rückkehr enthält **hl** die Anfangsadresse des Fileheaders.
a, **bc** und **de** enthalten weitere, dem Header entnommene Werte, die Sie aber auch dem Header selbst entnehmen können, da Sie ja dessen Startadresse haben.
Die Flags **carry** und **zero** geben Auskunft über den Erfolg der Aktion:
Carry=1 und zero=0 bedeuten, daß alles geklappt hat.
Carry=0 und zero=0 besagen, daß bereits noch ein File geöffnet ist.
Falls die ESC-Taste gedrückt wurde, ist carry=0 und zero=1.

CAS OUT OPEN öffnet ein Ausgabefile. Die Übergabeparameter und die Bedeutung der Flags sind die gleichen wie oben. Natürlich muß hier **de** die Adresse des Ausgabepuffers enthalten.

CAS IN CHAR holt ein Zeichen aus dem Eingabepuffer und übergibt es in **a**. War es das letzte Zeichen aus dem Puffer, wird automatisch ein neuer Block von der Kassette nachgezogen.
Wenn carry=0 und zero=0 sind, ist das Dateiende erreicht (EOF) oder das File war nicht geöffnet. Die anderen Kombination sind wie oben.

CAS OUT CHAR schreibt das Zeichen in **a** in den Ausgabepuffer. Ist dieser voll, wird er automatisch weggeschrieben.
Die Bedeutung der Flags ist wie oben.

```
******************************************************** CAS INITIALISE
2370  CD0124        call  2401         CAS IN ABANDON
2373  CD2E24        call  242E         CAS OUT ABANDON
2376  AF            xor   a
2377  CD8E23        call  238E         CAS NOISY
237A  214D01        ld    hl,014D
237D  3E19          ld    a,19

******************************************************** CAS SET SPEED
237F  29            add   hl,hl
2380  29            add   hl,hl
2381  29            add   hl,hl
2382  29            add   hl,hl
2383  29            add   hl,hl
2384  29            add   hl,hl
2385  0F            rrca
2386  0F            rrca
2387  E63F          and   3F
2389  6F            ld    l,a
238A  22D1B8        ld    (B8D1),hl    (Cass. Speed)
238D  C9            ret

******************************************************** CAS NOISY
238E  3200B8        ld    (B800),a     (Cass. Message Flag)
2391  C9            ret

******************************************************** CAS IN OPEN
2392  DD2102B8      ld    ix,B802      Input Buffer Status
2396  CDAF23        call  23AF         CAS Open
2399  D0            ret   nc
239A  E5            push  hl
239B  CD3F25        call  253F         File Header lesen
239E  ED5B1CB8      ld    de,(B81C)
23A2  ED4B1FB8      ld    bc,(B81F)
23A6  3A19B8        ld    a,(B819)
23A9  E1            pop   hl
23AA  C9            ret

******************************************************** CAS OUT OPEN
23AB  DD2147B8      ld    ix,B847      Output Buffer Status

******************************************************** CAS Open
23AF  DD7E00        ld    a,(ix+00)
23B2  B7            or    a
23B3  C0            ret   nz
23B4  DDE5          push  ix
23B6  E3            ex    (sp),hl
23B7  3601          ld    (hl),01
23B9  23            inc   hl
23BA  73            ld    (hl),e
23BB  23            inc   hl
23BC  72            ld    (hl),d
23BD  23            inc   hl
23BE  73            ld    (hl),e
```

```
23BF  23              inc   hl
23C0  72              ld    (hl),d
23C1  23              inc   hl
23C2  EB              ex    de,hl
23C3  E1              pop   hl
23C4  D5              push  de
23C5  0E40            ld    c,40
23C7  12              ld    (de),a
23C8  13              inc   de
23C9  0D              dec   c
23CA  20FB            jr    nz,23C7
23CC  D1              pop   de
23CD  D5              push  de
23CE  78              ld    a,b
23CF  FE10            cp    10
23D1  3802            jr    c,23D5
23D3  0610            ld    b,10
23D5  04              inc   b
23D6  48              ld    c,b
23D7  1807            jr    23E0
23D9  E7              rst   4
23DA  23              inc   hl
23DB  CDB627          call  27B6
23DE  12              ld    (de),a
23DF  13              inc   de
23E0  10F7            djnz  23D9
23E2  0D              dec   c
23E3  2809            jr    z,23EE
23E5  1B              dec   de
23E6  1A              ld    a,(de)
23E7  EE20            xor   20
23E9  2003            jr    nz,23EE
23EB  12              ld    (de),a
23EC  18F4            jr    23E2
23EE  E1              pop   hl
23EF  DD361501        ld    (ix+15),01
23F3  DD361716        ld    (ix+17),16
23F7  DD351C          dec   (ix+1C)
23FA  37              scf
23FB  C9              ret

*********************************************** CAS IN CLOSE
23FC  3A02B8          ld    a,(B802)      (Input Buffer Status)
23FF  B7              or    a
2400  C8              ret   z

********************************************** CAS IN ABANDON
2401  2102B8          ld    hl,B802       Input Buffer Status
2404  3E01            ld    a,01
2406  3600            ld    (hl),00
2408  23              inc   hl
2409  5E              ld    e,(hl)
240A  23              inc   hl
240B  56              ld    d,(hl)
240C  21CCB8          ld    hl,B8CC
```

```
240F  AE              xor   (hl)
2410  37              scf
2411  C0              ret   nz
2412  77              ld    (hl),a
2413  9F              sbc   a,a
2414  C9              ret

*********************************************** CAS OUT CLOSE
2415  3A47B8          ld    a,(B847)     (Output Buffer Status)
2418  FE04            cp    04
241A  2812            jr    z,242E       CAS OUT ABANDON
241C  C6FF            add   a,FF
241E  D0              ret   nc
241F  215DB8          ld    hl,B85D
2422  36FF            ld    (hl),FF
2424  23              inc   hl
2425  23              inc   hl
2426  7E              ld    a,(hl)
2427  23              inc   hl
2428  B6              or    (hl)
2429  37              scf
242A  C41426          call  nz,2614
242D  D0              ret   nc

********************************************** CAS OUT ABANDON
242E  2147B8          ld    hl,B847      Output Buffer Status
2431  3E02            ld    a,02
2433  18D1            jr    2406

************************************************ CAS IN CHAR
2435  E5              push  hl
2436  D5              push  de
2437  C5              push  bc
2438  0602            ld    b,02
243A  CD8B24          call  248B         Check Input Buffer Status
243D  201A            jr    nz,2459
243F  2A1AB8          ld    hl,(B81A)
2442  7C              ld    a,h
2443  B5              or    l
2444  37              scf
2445  CC3F25          call  z,253F       File Header lesen
2448  300F            jr    nc,2459
244A  2A1AB8          ld    hl,(B81A)
244D  2B              dec   hl
244E  221AB8          ld    (B81A),hl
2451  2A05B8          ld    hl,(B805)    (Pointer Input Buffer)
2454  E7              rst   4            ld a,(hl)
2455  23              inc   hl
2456  2205B8          ld    (B805),hl    (Pointer Input Buffer)
2459  182C            jr    2487
```

```
**************************************************** CAS OUT CHAR
245B  E5        push  hl
245C  D5        push  de
245D  C5        push  bc
245E  4F        ld    c,a
245F  2147B8    ld    hl,B847       Output Buffer Status
2462  0602      ld    b,02
2464  CD8E24    call  248E          Check Buffer Status
2467  201E      jr    nz,2487
2469  2A5FB8    ld    hl,(B85F)
246C  110008    ld    de,0800
246F  ED52      sbc   hl,de
2471  C5        push  bc
2472  D41426    call  nc,2614
2475  C1        pop   bc
2476  300F      jr    nc,2487
2478  2A5FB8    ld    hl,(B85F)
247B  23        inc   hl
247C  225FB8    ld    (B85F),hl
247F  2A4AB8    ld    hl,(B84A)     (Pointer Output Buffer)
2482  71        ld    (hl),c
2483  23        inc   hl
2484  224AB8    ld    (B84A),hl     (Pointer Output Buffer)
2487  C1        pop   bc
2488  D1        pop   de
2489  E1        pop   hl
248A  C9        ret

******************************************** Check Input Buffer Status
248B  2102B8    ld    hl,B802       Input Buffer Status

************************************************* Check Buffer Status
248E  7E        ld    a,(hl)
248F  B8        cp    b
2490  C8        ret   z
2491  EE01      xor   01
2493  C0        ret   nz
2494  70        ld    (hl),b
2495  C9        ret

**************************************************** CAS TEST EOF
2496  CD3524    call  2435          CAS IN CHAR
2499  D0        ret   nc

***************************************************** CAS RETURN
249A  E5        push  hl
249B  2A1AB8    ld    hl,(B81A)
249E  23        inc   hl
249F  221AB8    ld    (B81A),hl
24A2  2A05B8    ld    hl,(B805)     (Pointer Input Buffer)
24A5  2B        dec   hl
24A6  2205B8    ld    (B805),hl     (Pointer Input Buffer)
24A9  E1        pop   hl
24AA  C9        ret
```

```
****************************************************** CAS IN DIRECT
24AB  EB          ex     de,hl
24AC  0603        ld     b,03
24AE  CD8B24      call   248B            Check Input Buffer Status
24B1  C0          ret    nz
24B2  ED531CB8    ld     (B81C),de
24B6  CDCF24      call   24CF
24B9  2A1CB8      ld     hl,(B81C)
24BC  ED5B1AB8    ld     de,(B81A)
24C0  19          add    hl,de
24C1  221CB8      ld     (B81C),hl
24C4  CD3F25      call   253F            File Header lesen
24C7  38F0        jr     c,24B9
24C9  C8          ret    z
24CA  2AA6B8      ld     hl,(B8A6)
24CD  37          scf
24CE  C9          ret

24CF  2A03B8      ld     hl,(B803)       (Adr. Start Input Buffer)
24D2  ED5B1CB8    ld     de,(B81C)
24D6  ED4B1AB8    ld     bc,(B81A)
24DA  7B          ld     a,e
24DB  95          sub    l
24DC  7A          ld     a,d
24DD  9C          sbc    a,h
24DE  DAA6BA      jp     c,BAA6(0537)    KL LDIR CONT'D
24E1  09          add    hl,bc
24E2  2B          dec    hl
24E3  EB          ex     de,hl
24E4  09          add    hl,bc
24E5  2B          dec    hl
24E6  EB          ex     de,hl
24E7  C3ACBA      jp     BAAC(053D)      KL LDDR CONT'D

****************************************************** CAS OUT DIRECT
24EA  E5          push   hl
24EB  C5          push   bc
24EC  4F          ld     c,a
24ED  2147B8      ld     hl,B847         Output Buffer Status
24F0  0603        ld     b,03
24F2  CD8E24      call   248E            Check Buffer Status
24F5  79          ld     a,c
24F6  C1          pop    bc
24F7  E1          pop    hl
24F8  C0          ret    nz
24F9  325EB8      ld     (B85E),a
24FC  ED5364B8    ld     (B864),de
2500  ED4366B8    ld     (B866),bc
2504  2248B8      ld     (B848),hl       (Adr. Start Output Buffer)
2507  ED535FB8    ld     (B85F),de
250B  21FFF7      ld     hl,F7FF
250E  19          add    hl,de
250F  3F          ccf
2510  D8          ret    c
2511  210008      ld     hl,0800
```

```
2514  225FB8      ld    (B85F),hl
2517  EB          ex    de,hl
2518  ED52        sbc   hl,de
251A  E5          push  hl
251B  2A48B8      ld    hl,(B848)     (Adr. Start Output Buffer)
251E  19          add   hl,de
251F  E5          push  hl
2520  CD1426      call  2614
2523  E1          pop   hl
2524  D1          pop   de
2525  D0          ret   nc
2526  18DC        jr    2504

********************************************** CAS CATALOG
2528  2102B8      ld    hl,B802       Input Buffer Status
252B  7E          ld    a,(hl)
252C  B7          or    a
252D  C0          ret   nz
252E  3605        ld    (hl),05
2530  ED5303B8    ld    (B803),de     (Adr. Start Input Buffer)
2534  CD8E23      call  238E          CAS NOISY
2537  CD4425      call  2544
253A  38FB        jr    c,2537
253C  C30124      jp    2401          CAS IN ABANDON

********************************************** File Header lesen
253F  3A18B8      ld    a,(B818)
2542  B7          or    a
2543  C0          ret   nz
2544  010183      ld    bc,8301
2547  CD7326      call  2673
254A  305C        jr    nc,25A8
254C  218CB8      ld    hl,B88C
254F  114000      ld    de,0040
2552  3E2C        ld    a,2C
2554  CD3628      call  2836          CAS READ
2557  304F        jr    nc,25A8
2559  CDC525      call  25C5
255C  2057        jr    nz,25B5
255E  068B        ld    b,8B
2560  3802        jr    c,2564
2562  0689        ld    b,89
2564  CD9226      call  2692
2567  ED5B9FB8    ld    de,(B89F)
256B  2A1CB8      ld    hl,(B81C)
256E  3A02B8      ld    a,(B802)      (Input Buffer Status)
2571  FE03        cp    03
2573  280E        jr    z,2583
2575  21FFF7      ld    hl,F7FF
2578  19          add   hl,de
2579  3E04        ld    a,04
257B  382B        jr    c,25A8
257D  2A03B8      ld    hl,(B803)     (Adr. Start Input Buffer)
2580  2205B8      ld    (B805),hl     (Pointer Input Buffer)
2583  3E16        ld    a,16
```

```
2585  CD3628    call  2836        CAS READ
2588  301E      jr    nc,25A8
258A  2117B8    ld    hl,B817
258D  34        inc   (hl)
258E  3A9DB8    ld    a,(B89D)
2591  23        inc   hl
2592  77        ld    (hl),a
2593  AF        xor   a
2594  321EB8    ld    (B81E),a
2597  2A9FB8    ld    hl,(B89F)
259A  221AB8    ld    (B81A),hl
259D  CDBF27    call  27BF
25A0  3E8C      ld    a,8C
25A2  CC0C27    call  z,270C
25A5  37        scf
25A6  1865      jr    260D
25A8  B7        or    a
25A9  2102B8    ld    hl,B802     Input Buffer Status
25AC  285D      jr    z,260B
25AE  0685      ld    b,85
25B0  CD1327    call  2713
25B3  1897      jr    254C
25B5  F5        push  af
25B6  0688      ld    b,88
25B8  CD9226    call  2692
25BB  F1        pop   af
25BC  308E      jr    nc,254C
25BE  0687      ld    b,87
25C0  CD1127    call  2711
25C3  1887      jr    254C
25C5  CDBF27    call  27BF
25C8  37        scf
25C9  C8        ret   z
25CA  3A1EB8    ld    a,(B81E)
25CD  B7        or    a
25CE  281B      jr    z,25EB
25D0  3AA3B8    ld    a,(B8A3)
25D3  2F        cpl
25D4  B7        or    a
25D5  C0        ret   nz
25D6  3A07B8    ld    a,(B807)    (File Header Input)
25D9  B7        or    a
25DA  C4F325    call  nz,25F3
25DD  C0        ret   nz
25DE  218CB8    ld    hl,B88C
25E1  1107B8    ld    de,B807     File Header Input
25E4  014000    ld    bc,0040
25E7  EDB0      ldir
25E9  AF        xor   a
25EA  C9        ret

25EB  CDF325    call  25F3
25EE  C0        ret   nz
25EF  EB        ex    de,hl
25F0  1A        ld    a,(de)
```

```
25F1  BE              cp    (hl)
25F2  C9              ret

25F3  2107B8          ld    hl,B807        File Header Input
25F6  118CB8          ld    de,B88C
25F9  0610            ld    b,10
25FB  1A              ld    a,(de)
25FC  CDB627          call  27B6
25FF  4F              ld    c,a
2600  7E              ld    a,(hl)
2601  CDB627          call  27B6
2604  A9              xor   c
2605  C0              ret   nz
2606  23              inc   hl
2607  13              inc   de
2608  10F1            djnz  25FB
260A  C9              ret

260B  3604            ld    (hl),04
260D  9F              sbc   a,a
260E  F5              push  af
260F  CD4F2A          call  2A4F           CAS STOP MOTOR
2612  F1              pop   af
2613  C9              ret

2614  010284          ld    bc,8402
2617  CD7326          call  2673
261A  304A            jr    nc,2666
261C  068A            ld    b,8A
261E  114CB8          ld    de,B84C        File Header Output
2621  CD9526          call  2695
2624  2163B8          ld    hl,B863
2627  CD8826          call  2688
262A  303A            jr    nc,2666
262C  2A48B8          ld    hl,(B848)      (Adr. Start Output Buffer)
262F  224AB8          ld    (B84A),hl      (Pointer Output Buffer)
2632  2261B8          ld    (B861),hl
2635  E5              push  hl
2636  214CB8          ld    hl,B84C        File Header Output
2639  114000          ld    de,0040
263C  3E2C            ld    a,2C
263E  CD3F28          call  283F           CAS WRITE
2641  E1              pop   hl
2642  3022            jr    nc,2666
2644  ED5B5FB8        ld    de,(B85F)
2648  3E16            ld    a,16
264A  CD3F28          call  283F           CAS WRITE
264D  215DB8          ld    hl,B85D
2650  DC8826          call  c,2688
2653  3011            jr    nc,2666
2655  210000          ld    hl,0000
2658  225FB8          ld    (B85F),hl
265B  215CB8          ld    hl,B85C
265E  34              inc   (hl)
265F  AF              xor   a
```

```
2660  3263B8   ld    (B863),a
2663  37       scf
2664  18A7     jr    260D
2666  B7       or    a
2667  2147B8   ld    hl,B847      Output Buffer Status
266A  289F     jr    z,260B
266C  0686     ld    b,86
266E  CD1327   call  2713
2671  18B9     jr    262C
2673  21CCB8   ld    hl,B8CC
2676  79       ld    a,c
2677  BE       cp    (hl)
2678  3600     ld    (hl),00
267A  37       scf
267B  E5       push  hl
267C  C5       push  bc
267D  C46027   call  nz,2760
2680  C1       pop   bc
2681  E1       pop   hl
2682  9F       sbc   a,a
2683  D0       ret   nc
2684  71       ld    (hl),c
2685  C34B2A   jp    2A4B         CAS START MOTOR

2688  7E       ld    a,(hl)
2689  B7       or    a
268A  37       scf
268B  C8       ret   z
268C  012C01   ld    bc,012C
268F  C3722A   jp    2A72

2692  118CB8   ld    de,B88C
2695  3A00B8   ld    a,(B800)     (Cass. Message Flag)
2698  B7       or    a
2699  C0       ret   nz
269A  3201B8   ld    (B801),a
269D  CD8327   call  2783
26A0  CD2627   call  2726
26A3  1A       ld    a,(de)
26A4  B7       or    a
26A5  200A     jr    nz,26B1
26A7  3E8E     ld    a,8E
26A9  CD2727   call  2727
26AC  011000   ld    bc,0010
26AF  182E     jr    26DF
26B1  CDBF27   call  27BF
26B4  010010   ld    bc,1000
26B7  280D     jr    z,26C6
26B9  6B       ld    l,e
26BA  62       ld    h,d
26BB  7E       ld    a,(hl)
26BC  B7       or    a
26BD  2804     jr    z,26C3
26BF  0C       inc   c
26C0  23       inc   hl
```

```
26C1  10F8      djnz  26BB
26C3  78        ld    a,b
26C4  41        ld    b,c
26C5  4F        ld    c,a
26C6  CD8D27    call  278D
26C9  1A        ld    a,(de)
26CA  CDB627    call  27B6
26CD  B7        or    a
26CE  2002      jr    nz,26D2
26D0  3E20      ld    a,20
26D2  C5        push  bc
26D3  D5        push  de
26D4  CD3413    call  1334        TXT WR CHAR
26D7  D1        pop   de
26D8  C1        pop   bc
26D9  13        inc   de
26DA  10ED      djnz  26C9
26DC  CD5C27    call  275C
26DF  EB        ex    de,hl
26E0  09        add   hl,bc
26E1  EB        ex    de,hl
26E2  3E8D      ld    a,8D
26E4  CD2727    call  2727
26E7  0602      ld    b,02
26E9  CD8D27    call  278D
26EC  1A        ld    a,(de)
26ED  CDA427    call  27A4
26F0  CD5C27    call  275C
26F3  13        inc   de
26F4  CDBF27    call  27BF
26F7  200B      jr    nz,2704
26F9  13        inc   de
26FA  1A        ld    a,(de)
26FB  E60F      and   0F
26FD  C624      add   a,24
26FF  CD8027    call  2780        CAS Meldung (1 Zeichen) ausg.
2702  1858      jr    275C
2704  1A        ld    a,(de)
2705  2101B8    ld    hl,B801
2708  B6        or    (hl)
2709  C8        ret   z
270A  186F      jr    277B
270C  CD2727    call  2727
270F  186A      jr    277B
2711  3EFF      ld    a,FF
2713  F5        push  af
2714  CD1F27    call  271F        CAS Meldung (# in b) ausgeben
2717  F1        pop   af
2718  C660      add   a,60
271A  D48027    call  nc,2780     CAS Meldung (1 Zeichen) ausg.
271D  185C      jr    277B
```

```
****************************************** CAS Meldung (# in b) ausgeben
271F  CD8011    call   1180        TXT GET CURSOR
2722  25        dec    h
2723  C47B27    call   nz,277B
2726  78        ld     a,b
2727  E5        push   hl
2728  E67F      and    7F
272A  47        ld     b,a
272B  21C527    ld     hl,27C5     Kassetten-Meldungen
272E  2807      jr     z,2737
2730  7E        ld     a,(hl)
2731  23        inc    hl
2732  B7        or     a
2733  20FB      jr     nz,2730
2735  10F9      djnz   2730
2737  7E        ld     a,(hl)
2738  B7        or     a
2739  2805      jr     z,2740
273B  CD4327    call   2743
273E  18F7      jr     2737
2740  E1        pop    hl
2741  23        inc    hl
2742  C9        ret

2743  FA2727    jp     m,2727
2746  E5        push   hl
2747  0600      ld     b,00
2749  04        inc    b
274A  7E        ld     a,(hl)
274B  23        inc    hl
274C  07        rlca
274D  30FA      jr     nc,2749
274F  CD8D27    call   278D
2752  E1        pop    hl
2753  7E        ld     a,(hl)
2754  23        inc    hl
2755  E67F      and    7F
2757  CD8027    call   2780        CAS Meldung (1 Zeichen) ausg.
275A  10F7      djnz   2753
275C  3E20      ld     a,20
275E  1820      jr     2780        CAS Meldung (1 Zeichen) ausg.
2760  3A00B8    ld     a,(B800)    (Cass. Message Flag)
2763  B7        or     a
2764  37        scf
2765  C0        ret    nz
2766  CD1F27    call   271F        CAS Meldung (# in b) ausgeben
2769  CD421A    call   1A42        KM READ CHAR
276C  38FB      jr     c,2769
276E  CD7912    call   1279        TXT CUR ON
2771  CD561B    call   1B56        KM WAIT KEY
2774  CD8112    call   1281        TXT CUR OFF
2777  FE1B      cp     1B
2779  C8        ret    z
277A  37        scf
277B  CD8327    call   2783
```

```
277E  3E0A          ld    a,0A

****************************************** CAS Meldung (1 Zeichen) ausg.
2780  C30014        jp    1400          TXT OUTPUT

2783  F5            push  af
2784  E5            push  hl
2785  3E01          ld    a,01
2787  CD5E11        call  115E          TXT SET COLUMN
278A  E1            pop   hl
278B  F1            pop   af
278C  C9            ret

278D  D5            push  de
278E  CD5612        call  1256          TXT GET WINDOW
2791  5C            ld    e,h
2792  CD8011        call  1180          TXT GET CURSOR
2795  7C            ld    a,h
2796  3D            dec   a
2797  83            add   a,e
2798  80            add   a,b
2799  3D            dec   a
279A  BA            cp    d
279B  D1            pop   de
279C  D8            ret   c
279D  3EFF          ld    a,FF
279F  3201B8        ld    (B801),a
27A2  18D7          jr    277B
27A4  06FF          ld    b,FF
27A6  04            inc   b
27A7  D60A          sub   0A
27A9  30FB          jr    nc,27A6
27AB  C63A          add   a,3A
27AD  F5            push  af
27AE  78            ld    a,b
27AF  B7            or    a
27B0  C4A427        call  nz,27A4
27B3  F1            pop   af
27B4  18CA          jr    2780          CAS Meldung (1 Zeichen) ausg.
27B6  FE61          cp    61
27B8  D8            ret   c
27B9  FE7B          cp    7B
27BB  D0            ret   nc
27BC  C6E0          add   a,E0
27BE  C9            ret

27BF  3A02B8        ld    a,(B802)      (Input Buffer Status)
27C2  FE05          cp    05
27C4  C9            ret

************************************************ Kassetten-Meldungen
27C5  50 72 65 73 F3 00 50 4C           Press.PL
27CD  41 D9 74 68 65 EE 61 6E           AYthenan
27D5  F9 6B 65 79 BA 00 65 72           ykey:.er
27DD  72 6F F2 00 80 81 00 80           ror.....
```

```
27E5  52 45 C3 61 6E E4 81 00                        RECand..
27ED  52 65 61 E4 82 00 57 72                        Read..Wr
27F5  69 74 E5 82 00 52 65 77                        ite..rew
27FD  69 6E E4 74 61 70 E5 00                        indtape.
2805  46 6F 75 6E 64 20 A0 00                        Found  .
280D  4C 6F 61 64 69 6E E7 00                        Loading.
2815  53 61 76 69 6E E7 00 00                        Saving..
281D  4F EB 00 62 6C 6F 63 EB                        Ok.block
2825  00 55 6E 6E 61 6D 65 E4                        .Unnamed
282D  66 69 6C 65 20 20 20 A0                        file
2835  00                                             .

********************************************************** CAS READ

2836  CD7328         call   2873                    Motor Ein & Keyb. öffnen
2839  F5             push   af
283A  21B828         ld     hl,28B8
283D  1819           jr     2858

********************************************************** CAS WRITE
283F  CD7328         call   2873                    Motor Ein & Keyb. öffnen
2842  F5             push   af
2843  CD6429         call   2964
2846  21F728         ld     hl,28F7
2849  DC9D28         call   c,289D
284C  DC7929         call   c,2979
284F  180F           jr     2860

********************************************************** CAS CHECK
2851  CD7328         call   2873                    Motor Ein & Keyb. öffnen
2854  F5             push   af
2855  21C728         ld     hl,28C7
2858  E5             push   hl
2859  CD1929         call   2919
285C  E1             pop    hl
285D  DC9D28         call   c,289D
2860  D1             pop    de
2861  F5             push   af
2862  0182F7         ld     bc,F782                 Port A=Out
2865  ED49           out    (c),c
2867  0110F6         ld     bc,F610                 Motor ein
286A  ED49           out    (c),c
286C  FB             ei
286D  7A             ld     a,d
286E  CD512A         call   2A51                    CAS RESTORE MOTOR
2871  F1             pop    af
2872  C9             ret

**************************************************** Motor Ein & Keyb. öffnen
2873  32CDB8         ld     (B8CD),a
2876  1B             dec    de
2877  1C             inc    e
2878  E5             push   hl
2879  D5             push   de
287A  CD681E         call   1E68                    SOUND RESET
```

```
287D  D1          pop   de
287E  DDE1        pop   ix
2880  CD4B2A      call  2A4B            CAS START MOTOR
2883  F3          di
2884  010EF4      ld    bc,F40E         Sound I/O Port select
2887  ED49        out   (c),c
2889  01D0F6      ld    bc,F6D0         Strobe Ein
288C  ED49        out   (c),c
288E  0E10        ld    c,10            Strobe Aus
2890  ED49        out   (c),c
2892  0192F7      ld    bc,F792         Port A=In
2895  ED49        out   (c),c
2897  0158F6      ld    bc,F658         Keyb Y9 (ESC) öffnen
289A  ED49        out   (c),c           & Sound I/O auf Port A
289C  C9          ret

289D  7A          ld    a,d
289E  B7          or    a
289F  280D        jr    z,28AE
28A1  E5          push  hl
28A2  D5          push  de
28A3  1E00        ld    e,00
28A5  CDAE28      call  28AE
28A8  D1          pop   de
28A9  E1          pop   hl
28AA  D0          ret   nc
28AB  15          dec   d
28AC  20F3        jr    nz,28A1
28AE  01FFFF      ld    bc,FFFF
28B1  ED43D3B8    ld    (B8D3),bc
28B5  1601        ld    d,01
28B7  E9          jp    (hl)

28B8  CDB029      call  29B0
28BB  D0          ret   nc
28BC  DD7700      ld    (ix+00),a
28BF  DD23        inc   ix
28C1  15          dec   d
28C2  1D          dec   e
28C3  20F3        jr    nz,28B8
28C5  1812        jr    28D9
28C7  CDB029      call  29B0
28CA  D0          ret   nc
28CB  47          ld    b,a
28CC  CDDCBA      call  BADC (056D)     RAM LAM (IX)
28CF  A8          xor   b
28D0  3E03        ld    a,03
28D2  C0          ret   nz
28D3  DD23        inc   ix
28D5  15          dec   d
28D6  1D          dec   e
28D7  20EE        jr    nz,28C7
28D9  15          dec   d
28DA  2806        jr    z,28E2
28DC  CDB029      call  29B0
```

```
28DF  D0          ret   nc
28E0  18F7        jr    28D9
28E2  CDA629      call  29A6
28E5  CDB029      call  29B0
28E8  D0          ret   nc
28E9  AA          xor   d
28EA  2007        jr    nz,28F3
28EC  CDB029      call  29B0
28EF  D0          ret   nc
28F0  AB          xor   e
28F1  37          scf
28F2  C8          ret   z
28F3  3E02        ld    a,02
28F5  B7          or    a
28F6  C9          ret

28F7  CDDCBA      call  BADC (056D)    RAM LAM (IX)
28FA  CDF829      call  29F8
28FD  D0          ret   nc
28FE  DD23        inc   ix
2900  15          dec   d
2901  1D          dec   e
2902  20F3        jr    nz,28F7
2904  15          dec   d
2905  2807        jr    z,290E
2907  AF          xor   a
2908  CDF829      call  29F8
290B  D0          ret   nc
290C  18F6        jr    2904
290E  CDA629      call  29A6
2911  CDF829      call  29F8
2914  D0          ret   nc
2915  7B          ld    a,e
2916  C3F829      jp    29F8

2919  D5          push  de
291A  CD2329      call  2923
291D  D1          pop   de
291E  D8          ret   c
291F  B7          or    a
2920  C8          ret   z
2921  18F6        jr    2919
2923  2E55        ld    l,55
2925  CDCD29      call  29CD            CAS Input RD DATA & Test ESC
2928  D0          ret   nc
2929  110000      ld    de,0000
292C  62          ld    h,d
292D  CDCD29      call  29CD            CAS Input RD DATA & Test ESC
2930  D0          ret   nc
2931  EB          ex    de,hl
2932  0600        ld    b,00
2934  09          add   hl,bc
2935  EB          ex    de,hl
2936  25          dec   h
2937  20F4        jr    nz,292D
```

```
2939  61        ld    h,c
293A  79        ld    a,c
293B  92        sub   d
293C  4F        ld    c,a
293D  9F        sbc   a,a
293E  47        ld    b,a
293F  EB        ex    de,hl
2940  09        add   hl,bc
2941  EB        ex    de,hl
2942  CDCD29    call  29CD          CAS Input RD DATA & Test ESC
2945  D0        ret   nc
2946  7A        ld    a,d
2947  CB3F      srl   a
2949  CB3F      srl   a
294B  8A        adc   a,d
294C  94        sub   h
294D  38EA      jr    c,2939
294F  91        sub   c
2950  38E7      jr    c,2939
2952  7A        ld    a,d
2953  1F        rra
2954  8A        adc   a,d
2955  67        ld    h,a
2956  22CEB8    ld    (B8CE),hl
2959  CDB029    call  29B0
295C  D0        ret   nc
295D  21CDB8    ld    hl,B8CD
2960  AE        xor   (hl)
2961  C0        ret   nz
2962  37        scf
2963  C9        ret

2964  CD892A    call  2A89
2967  210108    ld    hl,0801
296A  CD7C29    call  297C
296D  D0        ret   nc
296E  B7        or    a
296F  CD082A    call  2A08
2972  D0        ret   nc
2973  3ACDB8    ld    a,(B8CD)
2976  C3F829    jp    29F8

2979  212100    ld    hl,0021
297C  06F4      ld    b,F4
297E  ED78      in    a,(c)
2980  E604      and   04
2982  C8        ret   z
2983  E5        push  hl
2984  37        scf
2985  CD082A    call  2A08
2988  E1        pop   hl
2989  2B        dec   hl
298A  7C        ld    a,h
298B  B5        or    l
298C  20EE      jr    nz,297C
```

```
298E  37              scf
298F  C9              ret

2990  2AD3B8          ld      hl,(B8D3)
2993  AC              xor     h
2994  F2A029          jp      p,29A0
2997  7C              ld      a,h
2998  EE08            xor     08
299A  67              ld      h,a
299B  7D              ld      a,l
299C  EE10            xor     10
299E  6F              ld      l,a
299F  37              scf
29A0  ED6A            adc     hl,hl
29A2  22D3B8          ld      (B8D3),hl
29A5  C9              ret

29A6  2AD3B8          ld      hl,(B8D3)
29A9  7D              ld      a,l
29AA  2F              cpl
29AB  5F              ld      e,a
29AC  7C              ld      a,h
29AD  2F              cpl
29AE  57              ld      d,a
29AF  C9              ret

29B0  D5              push    de
29B1  1E08            ld      e,08
29B3  2ACEB8          ld      hl,(B8CE)
29B6  CDD429          call    29D4
29B9  DCDD29          call    c,29DD
29BC  300D            jr      nc,29CB
29BE  7C              ld      a,h
29BF  91              sub     c
29C0  9F              sbc     a,a
29C1  CB12            rl      d
29C3  CD9029          call    2990
29C6  1D              dec     e
29C7  20EA            jr      nz,29B3
29C9  7A              ld      a,d
29CA  37              scf
29CB  D1              pop     de
29CC  C9              ret

****************************************** CAS Input RD DATA & Test ESC
29CD  06F4            ld      b,F4          Port A
29CF  ED78            in      a,(c)         Keyb X
29D1  E604            and     04            ESC ?
29D3  C8              ret     z             ja ⇒
29D4  ED5F            ld      a,r
29D6  C603            add     a,03
29D8  0F              rrca
29D9  0F              rrca
29DA  E61F            and     1F
29DC  4F              ld      c,a
```

```
29DD  06F5       ld    b,F5        Port B
29DF  79         ld    a,c
29E0  C602       add   a,02
29E2  4F         ld    c,a
29E3  380E       jr    c,29F3
29E5  ED78       in    a,(c)       Input RD DATA
29E7  AD         xor   l
29E8  E680       and   80
29EA  20F3       jr    nz,29DF
29EC  AF         xor   a
29ED  ED4F       ld    r,a
29EF  CB0D       rrc   l
29F1  37         scf
29F2  C9         ret

29F3  AF         xor   a
29F4  ED4F       ld    r,a
29F6  3C         inc   a
29F7  C9         ret

29F8  D5         push  de
29F9  1E08       ld    e,08
29FB  57         ld    d,a
29FC  CB02       rlc   d
29FE  CD082A     call  2A08
2A01  3003       jr    nc,2A06
2A03  1D         dec   e
2A04  20F6       jr    nz,29FC
2A06  D1         pop   de
2A07  C9         ret

2A08  ED4BD0B8   ld    bc,(B8D0)
2A0C  2AD2B8     ld    hl,(B8D2)
2A0F  9F         sbc   a,a
2A10  67         ld    h,a
2A11  2807       jr    z,2A1A
2A13  7D         ld    a,l
2A14  87         add   a,a
2A15  80         add   a,b
2A16  6F         ld    l,a
2A17  79         ld    a,c
2A18  90         sub   b
2A19  4F         ld    c,a
2A1A  7D         ld    a,l
2A1B  32D0B8     ld    (B8D0),a
2A1E  2E0A       ld    l,0A        WR DATA Aus
2A20  CD372A     call  2A37        CAS Output WR DATA
2A23  3806       jr    c,2A2B
2A25  91         sub   c
2A26  300C       jr    nc,2A34
2A28  2F         cpl
2A29  3C         inc   a
2A2A  4F         ld    c,a
2A2B  7C         ld    a,h
2A2C  CD9029     call  2990
```

```
2A2F  2E0B       ld    l,0B       WR DATA Ein
2A31  CD372A     call  2A37       CAS Output WR DATA
2A34  3E01       ld    a,01
2A36  C9         ret

********************************************** CAS Output WR DATA
2A37  ED5F       ld    a,r
2A39  CB3F       srl   a
2A3B  91         sub   c
2A3C  3003       jr    nc,2A41
2A3E  3C         inc   a
2A3F  20FD       jr    nz,2A3E
2A41  06F7       ld    b,F7       Port Control
2A43  ED69       out   (c),l      WR DATA
2A45  F5         push  af
2A46  AF         xor   a
2A47  ED4F       ld    r,a
2A49  F1         pop   af
2A4A  C9         ret

********************************************** CAS START MOTOR
2A4B  3E10       ld    a,10
2A4D  1802       jr    2A51       CAS RESTORE MOTOR

********************************************** CAS STOP MOTOR
2A4F  3EEF       ld    a,EF

********************************************** CAS RESTORE MOTOR
2A51  C5         push  bc
2A52  06F6       ld    b,F6       Port C
2A54  ED48       in    c,(c)
2A56  04         inc   b
2A57  E610       and   10
2A59  3E08       ld    a,08
2A5B  2801       jr    z,2A5E
2A5D  3C         inc   a
2A5E  ED79       out   (c),a      Motor Ein/Aus
2A60  37         scf
2A61  280C       jr    z,2A6F
2A63  79         ld    a,c
2A64  E610       and   10
2A66  C5         push  bc
2A67  01C800     ld    bc,00C8
2A6A  37         scf
2A6B  CC722A     call  z,2A72
2A6E  C1         pop   bc
2A6F  79         ld    a,c
2A70  C1         pop   bc
2A71  C9         ret

2A72  C5         push  bc
2A73  E5         push  hl
2A74  CD892A     call  2A89
2A77  3E42       ld    a,42
2A79  CDBD1C     call  1CBD       KM TEST KEY
```

```
2A7C  E1            pop   hl
2A7D  C1            pop   bc
2A7E  2007          jr    nz,2A87
2A80  0B            dec   bc
2A81  78            ld    a,b
2A82  B1            or    c
2A83  20ED          jr    nz,2A72
2A85  37            scf
2A86  C9            ret

2A87  AF            xor   a
2A88  C9            ret

2A89  018206        ld    bc,0682
2A8C  0B            dec   bc
2A8D  78            ld    a,b
2A8E  B1            or    c
2A8F  20FB          jr    nz,2A8C
2A91  C9            ret

2A92  C7            rst   0
2A93  C7            rst   0
2A94  C7            rst   0
2A95  C7            rst   0
2A96  C7            rst   0
2A97  C7            rst   0
```

2.5.10 SCREEN EDITOR (EDIT)

Bei dem Editor handelt es sich eigentlich gar nicht um ein Pack in dem Sinne, wie wir es bisher verstanden haben. Er wird vom Betriebssystem naämlich überhaupt nicht benutzt.
Vielmehr ist er im Zusammenhang mit den Arithmetik-Packs zu sehen. So wie diese, wird auch der Editor ausschließlich von BASIC angesprungen.

Uns fallen keine Routinen ein, die man einzeln nutzen könnte. Allenfalls den Editor als Ganzes.
Hierzu müssen Sie **hl** mit der Anfangsadresse Ihres zu editierenden Textes versorgen. Dieser Text darf nur max. 255 Zeichen enthalten, was auch der größten Länge einer BASIC-Zeile entspricht.

```
*************************************************************** EDIT
2A98  C5        push  bc
2A99  D5        push  de
2A9A  E5        push  hl
2A9B  E5        push  hl          Zeiger auf Eingabepuffer
2A9C  01FF00    ld    bc,00FF
2A9F  0C        inc   c           Zeichen im Puffer zählen
2AA0  7E        ld    a,(hl)
2AA1  23        inc   hl
2AA2  B7        or    a
2AA3  20FA      jr    nz,2A9F
2AA5  32DDB8    ld    (B8DD),a    (Insert Flag)
2AA8  CD6F2C    call  2C6F
2AAB  E1        pop   hl
2AAC  CD672D    call  2D67
2AAF  C5        push  bc
2AB0  E5        push  hl
2AB1  CDD92D    call  2DD9        Zeichen von Keyboard
2AB4  E1        pop   hl
2AB5  C1        pop   bc
2AB6  CDC62A    call  2AC6        EDIT Sprung ausführen
2AB9  30F4      jr    nc,2AAF
2ABB  F5        push  af
2ABC  CDD22C    call  2CD2
2ABF  F1        pop   af
2AC0  E1        pop   hl
2AC1  D1        pop   de
2AC2  C1        pop   bc
2AC3  FEFC      cp    FC
2AC5  C9        ret

********************************************** EDIT Sprung ausführen
2AC6  E5        push  hl
2AC7  21E02A    ld    hl,2AE0     EDIT Sprungtabelle 1
2ACA  5F        ld    e,a
2ACB  78        ld    a,b
2ACC  B1        or    c           Zeichen im Puffer ?
2ACD  7B        ld    a,e
2ACE  200B      jr    nz,2ADB     ja ⇨
2AD0  FEF0      cp    F0          eine der Cursor Tasten ?
2AD2  3807      jr    c,2ADB      nein ⇨
2AD4  FEF4      cp    F4          Cursor Taste & SHFT/CTRL ?
2AD6  3003      jr    nc,2ADB     ja ⇨
2AD8  211C2B    ld    hl,2B1C     EDIT Sprungtabelle 2
2ADB  CDF62D    call  2DF6        EDIT Sprungadr holen
2ADE  E3        ex    (sp),hl     ret-Adr. manipulieren
2ADF  C9        ret               Sprung gemäß Tabelle

*********************************************** EDIT Sprungtabelle 1
2AE0  13        db    13          Anzahl Einträge
2AE1  012C      dw    2C01        Zeichen einfügen
2AE3  FC        db    FC
2AE4  422B      dw    2B42        ESC
2AE6  EF        db    EF
```

```
2AE7  402B      dw   2B40      kein Effekt
2AE9  0D        db   0D
2AEA  692B      dw   2B69      ENTER
2AEC  F0        db   F0
2AED  B32B      dw   2BB3      CRSR UP (Puffer)
2AEF  F1        db   F1
2AF0  7E2B      dw   2B7E      CRSR DWN (Puffer)
2AF2  F2        db   F2
2AF3  AA2B      dw   2BAA      CRSR LEFT (Puffer)
2AF5  F3        db   F3
2AF6  752B      dw   2B75      CRSR RGHT (Puffer)
2AF8  F8        db   F8
2AF9  C72B      dw   2BC7      CTRL & CRSR UP
2AFB  F9        db   F9
2AFC  922B      dw   2B92      CTRL & CRSR DWN
2AFE  FA        db   FA
2AFF  BD2B      dw   2BBD      CTRL & CRSR LEFT
2B01  FB        db   FB
2B02  892B      dw   2B89      CTRL & CRSR RGHT
2B04  F4        db   F4
2B05  A22C      dw   2CA2      SHFT & CRSR UP
2B07  F5        db   F5
2B08  A72C      dw   2CA7      SHFT & CRSR DWN
2B0A  F6        db   F6
2B0B  9D2C      dw   2C9D      SHFT & CRSR LEFT
2B0D  F7        db   F7
2B0E  982C      dw   2C98      SHFT & CRSR RGHT
2B10  E0        db   E0
2B11  EA2C      dw   2CEA      COPY
2B13  7F        db   7F
2B14  3D2C      dw   2C3D      DEL
2B16  10        db   10
2B17  4A2C      dw   2C4A      CLR
2B19  E1        db   E1
2B1A  F92B      dw   2BF9      CTRL & TAB (Flip Insert)

********************************************* EDIT Sprungtabelle 2
2B1C  04        db   04        Anzahl Einträge
2B1D  2B2B      dw   2B2B      KLINGEL
2B1F  F0        db   F0
2B20  2F2B      dw   2B2F      CRSR UP
2B22  F1        db   F1
2B23  332B      dw   2B33      CRSR DWN
2B25  F2        db   F2
2B26  3B2B      dw   2B3B      CRSR LEFT
2B28  F3        db   F3
2B29  372B      dw   2B37      CRSR RGHT

*************************************************** KLINGEL
2B2B  3E07      ld   a,07      BEL
2B2D  180E      jr   2B3D
```

```
************************************************************ CRSR UP
2B2F  3E0B          ld     a,0B
2B31  180A          jr     2B3D

*********************************************************** CRSR DWN
2B33  3E0A          ld     a,0A
2B35  1806          jr     2B3D

********************************************************** CRSR RGHT
2B37  3E09          ld     a,09
2B39  1802          jr     2B3D

*********************************************************** CRSR LEFT
2B3B  3E08          ld     a,08
2B3D  CD0014        call   1400          TXT OUTPUT
2B40  B7            or     a
2B41  C9            ret

**************************************************************** ESC
2B42  F5            push   af
2B43  CD492B        call   2B49
2B46  F1            pop    af
2B47  37            scf
2B48  C9            ret

2B49  CD692B        call   2B69          ENTER
2B4C  21612B        ld     hl,2B61       *BREAK*-Meldung
2B4F  CD692B        call   2B69          ENTER
2B52  CD8011        call   1180          TXT GET CURSOR
2B55  25            dec    h
2B56  C8            ret    z
2B57  3E0D          ld     a,0D          CR
2B59  CD0014        call   1400          TXT OUTPUT
2B5C  3E0A          ld     a,0A          LF
2B5E  C30014        jp     1400          TXT OUTPUT

*********************************************************** *BREAK*-Meldung
2B61  2A 42 72 65 61 6B 2A 00            *Break*.

************************************************************ ENTER

2B69  F5            push   af
2B6A  7E            ld     a,(hl)
2B6B  23            inc    hl
2B6C  B7            or     a
2B6D  C4A82D        call   nz,2DA8
2B70  20F8          jr     nz,2B6A
2B72  F1            pop    af
2B73  37            scf
2B74  C9            ret
```

```
**********************************************  CRSR RGHT (Puffer)
2B75  1601      ld    d,01
2B77  CD932B    call  2B93
2B7A  CA2B2B    jp    z,2B2B          KLINGEL
2B7D  C9        ret

**********************************************  CRSR DWN (Puffer)
2B7E  CDEB2B    call  2BEB
2B81  79        ld    a,c
2B82  90        sub   b
2B83  BA        cp    d
2B84  DA2B2B    jp    c,2B2B          KLINGEL
2B87  180A      jr    2B93

**********************************************  CTRL & CRSR RGHT
2B89  CDEB2B    call  2BEB
2B8C  7A        ld    a,d
2B8D  93        sub   e
2B8E  C8        ret   z
2B8F  57        ld    d,a
2B90  1801      jr    2B93

**********************************************  CTRL & CRSR DWN
2B92  51        ld    d,c
2B93  78        ld    a,b
2B94  B9        cp    c
2B95  C8        ret   z
2B96  D5        push  de
2B97  CD502D    call  2D50
2B9A  7E        ld    a,(hl)
2B9B  D4A82D    call  nc,2DA8
2B9E  04        inc   b
2B9F  23        inc   hl
2BA0  D4672D    call  nc,2D67
2BA3  D1        pop   de
2BA4  15        dec   d
2BA5  20EC      jr    nz,2B93
2BA7  F6FF      or    FF
2BA9  C9        ret

**********************************************  CRSR LEFT (Puffer)
2BAA  1601      ld    d,01
2BAC  CDC82B    call  2BC8
2BAF  CA2B2B    jp    z,2B2B          KLINGEL
2BB2  C9        ret

**********************************************  CRSR UP (Puffer)
2BB3  CDEB2B    call  2BEB
2BB6  78        ld    a,b
2BB7  BA        cp    d
2BB8  DA2B2B    jp    c,2B2B          KLINGEL
2BBB  180B      jr    2BC8
```

```
*********************************************** CTRL & CRSR LEFT
2BBD  CDEB2B        call   2BEB
2BC0  7B            ld     a,e
2BC1  D601          sub    01
2BC3  C8            ret    z
2BC4  57            ld     d,a
2BC5  1801          jr     2BC8

************************************************ CTRL & CRSR UP
2BC7  51            ld     d,c
2BC8  78            ld     a,b
2BC9  B7            or     a
2BCA  C8            ret    z
2BCB  CD4A2D        call   2D4A
2BCE  3007          jr     nc,2BD7
2BD0  05            dec    b
2BD1  2B            dec    hl
2BD2  15            dec    d
2BD3  20F3          jr     nz,2BC8
2BD5  1811          jr     2BE8
2BD7  78            ld     a,b
2BD8  B7            or     a
2BD9  280A          jr     z,2BE5
2BDB  05            dec    b
2BDC  2B            dec    hl
2BDD  D5            push   de
2BDE  CD292D        call   2D29
2BE1  D1            pop    de
2BE2  15            dec    d
2BE3  20F2          jr     nz,2BD7
2BE5  CD672D        call   2D67
2BE8  F6FF          or     FF
2BEA  C9            ret

2BEB  E5            push   hl
2BEC  CD5612        call   1256          TXT GET WINDOW
2BEF  7A            ld     a,d
2BF0  94            sub    h
2BF1  3C            inc    a
2BF2  57            ld     d,a
2BF3  CD8011        call   1180          TXT GET CURSOR
2BF6  5C            ld     e,h
2BF7  E1            pop    hl
2BF8  C9            ret

******************************************* CTRL & TAB (Flip Insert)
2BF9  3ADDB8        ld     a,(B8DD)      (Insert Flag)
2BFC  2F            cpl
2BFD  32DDB8        ld     (B8DD),a      (Insert Flag)
2C00  C9            ret
```

```
************************************************ Zeichen einfügen
2C01  B7            or    a
2C02  C8            ret   z
2C03  5F            ld    e,a
2C04  3ADDB8        ld    a,(B8DD)        (Insert Flag)
2C07  B7            or    a
2C08  280D          jr    z,2C17
2C0A  78            ld    a,b
2C0B  B9            cp    c
2C0C  2809          jr    z,2C17
2C0E  73            ld    (hl),e
2C0F  7B            ld    a,e
2C10  CDA82D        call  2DA8
2C13  23            inc   hl
2C14  04            inc   b
2C15  B7            or    a
2C16  C9            ret

2C17  79            ld    a,c
2C18  FEFF          cp    FF
2C1A  CA2B2B        jp    z,2B2B          KLINGEL
2C1D  AF            xor   a
2C1E  32DCB8        ld    (B8DC),a
2C21  7B            ld    a,e
2C22  CDA82D        call  2DA8
2C25  0C            inc   c
2C26  E5            push  hl
2C27  7E            ld    a,(hl)
2C28  73            ld    (hl),e
2C29  5F            ld    e,a
2C2A  23            inc   hl
2C2B  B7            or    a
2C2C  20F9          jr    nz,2C27
2C2E  77            ld    (hl),a
2C2F  E1            pop   hl
2C30  04            inc   b
2C31  23            inc   hl
2C32  CD672D        call  2D67
2C35  3ADCB8        ld    a,(B8DC)
2C38  B7            or    a
2C39  C4292D        call  nz,2D29
2C3C  C9            ret

*********************************************************** DEL
2C3D  78            ld    a,b
2C3E  B7            or    a
2C3F  CA2B2B        jp    z,2B2B          KLINGEL
2C42  CD4A2D        call  2D4A
2C45  D22B2B        jp    nc,2B2B         KLINGEL
2C48  05            dec   b
2C49  2B            dec   hl
```

```
*************************************************************************** CLR
2C4A  78          ld     a,b
2C4B  B9          cp     c
2C4C  CA2B2B      jp     z,2B2B        KLINGEL
2C4F  E5          push   hl
2C50  23          inc    hl
2C51  7E          ld     a,(hl)
2C52  2B          dec    hl
2C53  77          ld     (hl),a
2C54  23          inc    hl
2C55  B7          or     a
2C56  20F8        jr     nz,2C50
2C58  2B          dec    hl
2C59  3620        ld     (hl),20
2C5B  32DCB8      ld     (B8DC),a
2C5E  E3          ex     (sp),hl
2C5F  CD672D      call   2D67
2C62  E3          ex     (sp),hl
2C63  3600        ld     (hl),00
2C65  E1          pop    hl
2C66  0D          dec    c
2C67  3ADCB8      ld     a,(B8DC)
2C6A  B7          or     a
2C6B  C42D2D      call   nz,2D2D
2C6E  C9          ret

2C6F  210000      ld     hl,0000
2C72  22DEB8      ld     (B8DE),hl
2C75  C9          ret

2C76  ED5BDEB8    ld     de,(B8DE)
2C7A  7C          ld     a,h
2C7B  AA          xor    d
2C7C  C0          ret    nz
2C7D  7D          ld     a,l
2C7E  AB          xor    e
2C7F  C0          ret    nz
2C80  37          scf
2C81  C9          ret

2C82  4F          ld     c,a
2C83  2ADEB8      ld     hl,(B8DE)
2C86  7C          ld     a,h
2C87  B5          or     l
2C88  C8          ret    z
2C89  7D          ld     a,l
2C8A  81          add    a,c
2C8B  6F          ld     l,a
2C8C  CDCE11      call   11CE          TXT VALIDATE
2C8F  3803        jr     c,2C94
2C91  210000      ld     hl,0000
2C94  22DEB8      ld     (B8DE),hl
2C97  C9          ret
```

```
*********************************************** SHFT & CRSR RGHT
2C98  110001        ld    de,0100
2C9B  180D          jr    2CAA

************************************************ SHFT & CRSR LEFT
2C9D  1100FF        ld    de,FF00
2CA0  1808          jr    2CAA

************************************************* SHFT & CRSR UP
2CA2  11FF00        ld    de,00FF
2CA5  1803          jr    2CAA

************************************************ SHFT & CRSR DWN
2CA7  110100        ld    de,0001
2CAA  C5            push  bc
2CAB  E5            push  hl
2CAC  2ADEB8        ld    hl,(B8DE)
2CAF  7C            ld    a,h
2CB0  B5            or    l
2CB1  CC8011        call  z,1180      TXT GET CURSOR
2CB4  7C            ld    a,h
2CB5  82            add   a,d
2CB6  67            ld    h,a
2CB7  7D            ld    a,l
2CB8  83            add   a,e
2CB9  6F            ld    l,a
2CBA  CDCE11        call  11CE        TXT VALIDATE
2CBD  300B          jr    nc,2CCA
2CBF  E5            push  hl
2CC0  CDD22C        call  2CD2
2CC3  E1            pop   hl
2CC4  22DEB8        ld    (B8DE),hl
2CC7  CDCD2C        call  2CCD
2CCA  E1            pop   hl
2CCB  C1            pop   bc
2CCC  C9            ret

2CCD  116812        ld    de,1268     TXT PLACE/REMOVE CURSOR
2CD0  1803          jr    2CD5
2CD2  116812        ld    de,1268     TXT PLACE/REMOVE CURSOR
2CD5  2ADEB8        ld    hl,(B8DE)
2CD8  7C            ld    a,h
2CD9  B5            or    l
2CDA  C8            ret   z
2CDB  E5            push  hl
2CDC  CD8011        call  1180        TXT GET CURSOR
2CDF  E3            ex    (sp),hl
2CE0  CD7411        call  1174        TXT SET CURSOR
2CE3  CD1600        call  0016
2CE6  E1            pop   hl
2CE7  C37411        jp    1174        TXT SET CURSOR
```

```
****************************************************** COPY
2CEA  C5          push  bc
2CEB  E5          push  hl
2CEC  CD8011      call  1180          TXT GET CURSOR
2CEF  EB          ex    de,hl
2CF0  2ADEB8      ld    hl,(B8DE)
2CF3  7C          ld    a,h
2CF4  B5          or    l
2CF5  200C        jr    nz,2D03
2CF7  78          ld    a,b
2CF8  B1          or    c
2CF9  2026        jr    nz,2D21
2CFB  CD8011      call  1180          TXT GET CURSOR
2CFE  22DEB8      ld    (B8DE),hl
2D01  1806        jr    2D09
2D03  CD7411      call  1174          TXT SET CURSOR
2D06  CD6812      call  1268          TXT PLACE/REMOVE CURSOR
2D09  CDAB13      call  13AB          TXT RD CHAR
2D0C  F5          push  af
2D0D  EB          ex    de,hl
2D0E  CD7411      call  1174          TXT SET CURSOR
2D11  2ADEB8      ld    hl,(B8DE)
2D14  24          inc   h
2D15  CDCE11      call  11CE          TXT VALIDATE
2D18  3003        jr    nc,2D1D
2D1A  22DEB8      ld    (B8DE),hl
2D1D  CDCD2C      call  2CCD
2D20  F1          pop   af
2D21  E1          pop   hl
2D22  C1          pop   bc
2D23  DA012C      jp    c,2C01        Zeichen einfügen
2D26  C32B2B      jp    2B2B          KLINGEL

2D29  1601        ld    d,01
2D2B  1802        jr    2D2F
2D2D  16FF        ld    d,FF
2D2F  C5          push  bc
2D30  E5          push  hl
2D31  D5          push  de
2D32  CDD22C      call  2CD2
2D35  D1          pop   de
2D36  2ADEB8      ld    hl,(B8DE)
2D39  7C          ld    a,h
2D3A  B5          or    l
2D3B  2809        jr    z,2D46
2D3D  7C          ld    a,h
2D3E  82          add   a,d
2D3F  67          ld    h,a
2D40  CD8C2C      call  2C8C
2D43  CDCD2C      call  2CCD
2D46  E1          pop   hl
2D47  C1          pop   bc
2D48  B7          or    a
2D49  C9          ret
```

```
2D4A  D5          push   de
2D4B  1108FF      ld     de,FF08
2D4E  1804        jr     2D54
2D50  D5          push   de
2D51  110901      ld     de,0109
2D54  C5          push   bc
2D55  E5          push   hl
2D56  CD8011      call   1180        TXT GET CURSOR
2D59  7A          ld     a,d
2D5A  84          add    a,h
2D5B  67          ld     h,a
2D5C  CDCE11      call   11CE        TXT VALIDATE
2D5F  7B          ld     a,e
2D60  DC0014      call   c,1400      TXT OUTPUT
2D63  E1          pop    hl
2D64  C1          pop    bc
2D65  D1          pop    de
2D66  C9          ret

2D67  C5          push   bc
2D68  E5          push   hl
2D69  EB          ex     de,hl
2D6A  CD8011      call   1180        TXT GET CURSOR
2D6D  4F          ld     c,a
2D6E  EB          ex     de,hl
2D6F  7E          ld     a,(hl)
2D70  23          inc    hl
2D71  B7          or     a
2D72  C4852D      call   nz,2D85
2D75  20F8        jr     nz,2D6F
2D77  CD8011      call   1180        TXT GET CURSOR
2D7A  91          sub    c
2D7B  EB          ex     de,hl
2D7C  85          add    a,l
2D7D  6F          ld     l,a
2D7E  CD7411      call   1174        TXT SET CURSOR
2D81  E1          pop    hl
2D82  C1          pop    bc
2D83  B7          or     a
2D84  C9          ret

2D85  F5          push   af
2D86  C5          push   bc
2D87  D5          push   de
2D88  E5          push   hl
2D89  47          ld     b,a
2D8A  CD8011      call   1180        TXT GET CURSOR
2D8D  91          sub    c
2D8E  83          add    a,e
2D8F  5F          ld     e,a
2D90  48          ld     c,b
2D91  CDCE11      call   11CE        TXT VALIDATE
2D94  3805        jr     c,2D9B
2D96  78          ld     a,b
2D97  87          add    a,a
```

```
2D98  3C          inc   a
2D99  83          add   a,e
2D9A  5F          ld    e,a
2D9B  EB          ex    de,hl
2D9C  CDCE11      call  11CE        TXT VALIDATE
2D9F  79          ld    a,c
2DA0  DCA82D      call  c,2DA8
2DA3  E1          pop   hl
2DA4  D1          pop   de
2DA5  C1          pop   bc
2DA6  F1          pop   af
2DA7  C9          ret

2DA8  F5          push  af
2DA9  C5          push  bc
2DAA  D5          push  de
2DAB  E5          push  hl
2DAC  47          ld    b,a
2DAD  CD8011      call  1180        TXT GET CURSOR
2DB0  4F          ld    c,a
2DB1  C5          push  bc
2DB2  CDCE11      call  11CE        TXT VALIDATE
2DB5  C1          pop   bc
2DB6  DC762C      call  c,2C76
2DB9  F5          push  af
2DBA  DCD22C      call  c,2CD2
2DBD  78          ld    a,b
2DBE  C5          push  bc
2DBF  CD3413      call  1334        TXT WR CHAR
2DC2  C1          pop   bc
2DC3  CD8011      call  1180        TXT GET CURSOR
2DC6  91          sub   c
2DC7  C4822C      call  nz,2C82
2DCA  F1          pop   af
2DCB  3007        jr    nc,2DD4
2DCD  9F          sbc   a,a
2DCE  32DCB8      ld    (B8DC),a
2DD1  CDCD2C      call  2CCD
2DD4  E1          pop   hl
2DD5  D1          pop   de
2DD6  C1          pop   bc
2DD7  F1          pop   af
2DD8  C9          ret

********************************************* Zeichen von Keyboard
2DD9  CD8011      call  1180        TXT GET CURSOR
2DDC  4F          ld    c,a
2DDD  CDCE11      call  11CE        TXT VALIDATE
2DE0  CD762C      call  2C76
2DE3  DA3C1A      jp    c,1A3C      KM WAIT CHAR
2DE6  CD7912      call  1279        TXT CUR ON
2DE9  CD8011      call  1180        TXT GET CURSOR
2DEC  91          sub   c
2DED  C4822C      call  nz,2C82
2DF0  CD3C1A      call  1A3C        KM WAIT CHAR
```

```
2DF3  C38112           jp     1281          TXT CUR OFF

********************************************** EDIT Sprungadr holen
2DF6  F5               push   af
2DF7  C5               push   bc
2DF8  46               ld     b,(hl)
2DF9  23               inc    hl
2DFA  E5               push   hl
2DFB  23               inc    hl
2DFC  23               inc    hl
2DFD  BE               cp     (hl)
2DFE  23               inc    hl
2DFF  2804             jr     z,2E05
2E01  05               dec    b
2E02  20F7             jr     nz,2DFB
2E04  E3               ex     (sp),hl
2E05  F1               pop    af
2E06  7E               ld     a,(hl)
2E07  23               inc    hl
2E08  66               ld     h,(hl)
2E09  6F               ld     l,a
2E0A  C1               pop    bc
2E0B  F1               pop    af
2E0C  C9               ret

2E0D  C7               rst    0
2E0E  C7               rst    0
2E0F  C7               rst    0
2E10  C7               rst    0
2E11  C7               rst    0
2E12  C7               rst    0
2E13  C7               rst    0
2E14  C7               rst    0
2E15  C7               rst    0
2E16  C7               rst    0
2E17  C7               rst    0
```

2.6 Der Character – Generator

Nicht, daß wir Sie mit den folgenden Seiten langweilen wollen oder wir der Meinung sind, das Buch sei noch nicht umfangreich genug.
Was wir glauben, ist einfach, daß der Zeichensatz ein wichtiges Betriebsmittel ist, welchem sogar im BASIC – Befehlsvorrat eigene Kommandos zugestanden werden.
Damit Sie bei deren Anwendung nicht jedesmal das Rad neu erfinden müssen, z.B. bei der Erzeugung von Umlauten, brauchen Sie sich nur das 'a' heraussuchen und die beiden Pünktchen darübersetzen. Die so gefundenen Werte setzen Sie in Ihren Befehl ein.
Warum es wichtig ist, daß Sie sich möglichst an den bereits vorhandenen Zeichen orientieren, ist schnell erklärt:
Es fällt Ihnen sicher auf, daß alle vertikalen Linienelemente aus wenigstens zwei benachbarten Punkten zusammengesetzt sind. Der Grund dafür ist die Tatsache, daß Sie ein einziges Pixel allein auf Ihrem Bildschirm schwerlich wiederfinden würden, auf dem Farbbildschirm noch schlechter als auf dem Grünmonitor, weil dort noch die Schlitzmaske im Wege ist, auf einen deren Stege dieser eine Punkt ja zufällig treffen könnte.
Ziehen Sie also hieraus die Erfahrung, bei senkrechten Linien immer ein Pärchen vorzusehen.

So, nun steht Ihren eigenen Versuchen nichts mehr im Wege.

Address	Value
3800	FF
3801	C3
3802	C3
3803	C3
3804	C3
3805	C3
3806	C3
3807	FF

Address	Value
3808	FF
3809	C0
380A	C0
380B	C0
380C	C0
380D	C0
380E	C0
380F	C0

Address	Value
3810	18
3811	18
3812	18
3813	18
3814	18
3815	18
3816	18
3817	FF

Address	Value
3818	03
3819	03
381A	03
381B	03
381C	03
381D	03
381E	03
381F	FF

Address	Value
3820	0C
3821	18
3822	30
3823	7E
3824	0C
3825	18
3826	30
3827	00

Address	Value
3828	FF
3829	C3
382A	E7
382B	DB
382C	DB
382D	E7
382E	C3
382F	FF

Address	Value
3830	00
3831	01
3832	03
3833	06
3834	CC
3835	78
3836	30
3837	00

Address	Value
3838	3C
3839	66
383A	C3
383B	C3
383C	FF
383D	24
383E	E7
383F	00

Address	Value
3840	00
3841	00
3842	30
3843	60
3844	FF
3845	60
3846	30
3847	00

Address	Value
3848	00
3849	00
384A	0C
384B	06
384C	FF
384D	06
384E	0C
384F	00

Address	Value
3850	18
3851	18
3852	18
3853	18
3854	DB
3855	7E
3856	3C
3857	18

Address	Value
3858	18
3859	3C
385A	7E
385B	DB
385C	18
385D	18
385E	18
385F	18

Address	Value
3860	18
3861	5A
3862	3C
3863	99
3864	DB
3865	7E
3866	3C
3867	18
3868	00
3869	03
386A	33
386B	63
386C	FE
386D	60
386E	30
386F	00
3870	3C
3871	66
3872	FF
3873	DB
3874	DB
3875	FF
3876	66
3877	3C
3878	3C
3879	66
387A	C3
387B	DB
387C	DB
387D	C3
387E	66
387F	3C
3880	FF
3881	C3
3882	C3
3883	FF
3884	C3
3885	C3
3886	C3
3887	FF
3888	3C
3889	7E
388A	DB
388B	DB
388C	DF
388D	C3
388E	66
388F	3C
3890	3C
3891	66
3892	C3
3893	DF
3894	DB
3895	DB
3896	7E
3897	3C
3898	3C
3899	66
389A	C3
389B	FB
389C	DB
389D	DB
389E	7E
389F	3C
38A0	3C
38A1	7E
38A2	DB
38A3	DB
38A4	FB
38A5	C3
38A6	66
38A7	3C
38A8	00
38A9	01
38AA	33
38AB	1E
38AC	CE
38AD	7B
38AE	31
38AF	00
38B0	7E
38B1	66
38B2	66
38B3	66
38B4	66
38B5	66
38B6	66
38B7	E7
38B8	03
38B9	03
38BA	03
38BB	FF
38BC	03
38BD	03
38BE	03
38BF	00

Address	Value
38C0	FF
38C1	66
38C2	3C
38C3	18
38C4	18
38C5	3C
38C6	66
38C7	FF

Address	Value
38C8	18
38C9	18
38CA	3C
38CB	3C
38CC	3C
38CD	3C
38CE	18
38CF	18

Address	Value
38D0	3C
38D1	66
38D2	66
38D3	30
38D4	18
38D5	00
38D6	18
38D7	00

Address	Value
38D8	3C
38D9	66
38DA	C3
38DB	FF
38DC	C3
38DD	C3
38DE	66
38DF	3C

Address	Value
38E0	FF
38E1	DB
38E2	DB
38E3	DB
38E4	FB
38E5	C3
38E6	C3
38E7	FF

Address	Value
38E8	FF
38E9	C3
38EA	C3
38EB	FB
38EC	DB
38ED	DB
38EE	DB
38EF	FF

Address	Value
38F0	FF
38F1	C3
38F2	C3
38F3	DF
38F4	DB
38F5	DB
38F6	DB
38F7	FF

Address	Value
38F8	FF
38F9	DB
38FA	DB
38FB	DB
38FC	DF
38FD	C3
38FE	C3
38FF	FF

Address	Value
3900	00
3901	00
3902	00
3903	00
3904	00
3905	00
3906	00
3907	00

Address	Value
3908	18
3909	18
390A	18
390B	18
390C	18
390D	00
390E	18
390F	00

Address	Value
3910	6C
3911	6C
3912	6C
3913	00
3914	00
3915	00
3916	00
3917	00

Address	Value
3918	6C
3919	6C
391A	FE
391B	6C
391C	FE
391D	6C
391E	6C
391F	00

3920	18
3921	3E
3922	58
3923	3C
3924	1A
3925	7C
3926	18
3927	00

3928	00
3929	C6
392A	CC
392B	18
392C	30
392D	66
392E	C6
392F	00

3930	38
3931	6C
3932	38
3933	76
3934	DC
3935	CC
3936	76
3937	00

3938	18
3939	18
393A	30
393B	00
393C	00
393D	00
393E	00
393F	00

3940	0C
3941	18
3942	30
3943	30
3944	30
3945	18
3946	0C
3947	00

3948	30
3949	18
394A	0C
394B	0C
394C	0C
394D	18
394E	30
394F	00

3950	00
3951	66
3952	3C
3953	FF
3954	3C
3955	66
3956	00
3957	00

3958	00
3959	18
395A	18
395B	7E
395C	18
395D	18
395E	00
395F	00

3960	00
3961	00
3962	00
3963	00
3964	00
3965	18
3966	18
3967	30

3968	00
3969	00
396A	00
396B	7E
396C	00
396D	00
396E	00
396F	00

3970	00
3971	00
3972	00
3973	00
3974	00
3975	18
3976	18
3977	00

3978	06
3979	0C
397A	18
397B	30
397C	60
397D	C0
397E	80
397F	00

Address	Data
3980	7C
3981	C6
3982	CE
3983	D6
3984	E6
3985	C6
3986	7C
3987	00
3988	18
3989	38
398A	18
398B	18
398C	18
398D	18
398E	7E
398F	00
3990	3C
3991	66
3992	06
3993	3C
3994	60
3995	66
3996	7E
3997	00
3998	3C
3999	66
399A	06
399B	1C
399C	06
399D	66
399E	3C
399F	00
39A0	1C
39A1	3C
39A2	6C
39A3	CC
39A4	FE
39A5	0C
39A6	1E
39A7	00
39A8	7E
39A9	62
39AA	60
39AB	7C
39AC	06
39AD	66
39AE	3C
39AF	00
39B0	3C
39B1	66
39B2	60
39B3	7C
39B4	66
39B5	66
39B6	3C
39B7	00
39B8	7E
39B9	66
39BA	06
39BB	0C
39BC	18
39BD	18
39BE	18
39BF	00
39C0	3C
39C1	66
39C2	66
39C3	3C
39C4	66
39C5	66
39C6	3C
39C7	00
39C8	3C
39C9	66
39CA	66
39CB	3E
39CC	06
39CD	66
39CE	3C
39CF	00
39D0	00
39D1	00
39D2	18
39D3	18
39D4	00
39D5	18
39D6	18
39D7	00
39D8	00
39D9	00
39DA	18
39DB	18
39DC	00
39DD	18
39DE	18
39DF	30

Address	Value
39E0	0C
39E1	18
39E2	30
39E3	60
39E4	30
39E5	18
39E6	0C
39E7	00
39E8	00
39E9	00
39EA	7E
39EB	00
39EC	00
39ED	7E
39EE	00
39EF	00
39F0	60
39F1	30
39F2	18
39F3	0C
39F4	18
39F5	30
39F6	60
39F7	00
39F8	3C
39F9	66
39FA	66
39FB	0C
39FC	18
39FD	00
39FE	18
39FF	00
3A00	7C
3A01	C6
3A02	DE
3A03	DE
3A04	DE
3A05	C0
3A06	7C
3A07	00
3A08	18
3A09	3C
3A0A	66
3A0B	66
3A0C	7E
3A0D	66
3A0E	66
3A0F	00
3A10	FC
3A11	66
3A12	66
3A13	7C
3A14	66
3A15	66
3A16	FC
3A17	00
3A18	3C
3A19	66
3A1A	C0
3A1B	C0
3A1C	C0
3A1D	66
3A1E	3C
3A1F	00
3A20	F8
3A21	6C
3A22	66
3A23	66
3A24	66
3A25	6C
3A26	F8
3A27	00
3A28	FE
3A29	62
3A2A	68
3A2B	78
3A2C	68
3A2D	62
3A2E	FE
3A2F	00
3A30	FE
3A31	62
3A32	68
3A33	78
3A34	68
3A35	60
3A36	F0
3A37	00
3A38	3C
3A39	66
3A3A	C0
3A3B	C0
3A3C	CE
3A3D	66
3A3E	3E
3A3F	00

Address	Value
3A40	66
3A41	66
3A42	66
3A43	7E
3A44	66
3A45	66
3A46	66
3A47	00
3A48	7E
3A49	18
3A4A	18
3A4B	18
3A4C	18
3A4D	18
3A4E	7E
3A4F	00
3A50	1E
3A51	0C
3A52	0C
3A53	0C
3A54	CC
3A55	CC
3A56	78
3A57	00
3A58	E6
3A59	66
3A5A	6C
3A5B	78
3A5C	6C
3A5D	66
3A5E	E6
3A5F	00
3A60	F0
3A61	60
3A62	60
3A63	60
3A64	62
3A65	66
3A66	FE
3A67	00
3A68	C6
3A69	EE
3A6A	FE
3A6B	FE
3A6C	D6
3A6D	C6
3A6E	C6
3A6F	00
3A70	C6
3A71	E6
3A72	F6
3A73	DE
3A74	CE
3A75	C6
3A76	C6
3A77	00
3A78	38
3A79	6C
3A7A	C6
3A7B	C6
3A7C	C6
3A7D	6C
3A7E	38
3A7F	00
3A80	FC
3A81	66
3A82	66
3A83	7C
3A84	60
3A85	60
3A86	F0
3A87	00
3A88	38
3A89	6C
3A8A	C6
3A8B	C6
3A8C	DA
3A8D	CC
3A8E	76
3A8F	00
3A90	FC
3A91	66
3A92	66
3A93	7C
3A94	6C
3A95	66
3A96	E6
3A97	00
3A98	3C
3A99	66
3A9A	60
3A9B	3C
3A9C	06
3A9D	66
3A9E	3C
3A9F	00

Address	Value	Address	Value
3AA0	7E	3AA8	66
3AA1	5A	3AA9	66
3AA2	18	3AAA	66
3AA3	18	3AAB	66
3AA4	18	3AAC	66
3AA5	18	3AAD	66
3AA6	3C	3AAE	3C
3AA7	00	3AAF	00
3AB0	66	3AB8	C6
3AB1	66	3AB9	C6
3AB2	66	3ABA	C6
3AB3	66	3ABB	D6
3AB4	66	3ABC	FE
3AB5	3C	3ABD	EE
3AB6	18	3ABE	C6
3AB7	00	3ABF	00
3AC0	C6	3AC8	66
3AC1	6C	3AC9	66
3AC2	38	3ACA	66
3AC3	38	3ACB	3C
3AC4	6C	3ACC	18
3AC5	C6	3ACD	18
3AC6	C6	3ACE	3C
3AC7	00	3ACF	00
3AD0	FE	3AD8	3C
3AD1	C6	3AD9	30
3AD2	8C	3ADA	30
3AD3	18	3ADB	30
3AD4	32	3ADC	30
3AD5	66	3ADD	30
3AD6	FE	3ADE	3C
3AD7	00	3ADF	00
3AE0	C0	3AE8	3C
3AE1	60	3AE9	0C
3AE2	30	3AEA	0C
3AE3	18	3AEB	0C
3AE4	0C	3AEC	0C
3AE5	06	3AED	0C
3AE6	02	3AEE	3C
3AE7	00	3AEF	00
3AF0	18	3AF8	00
3AF1	3C	3AF9	00
3AF2	7E	3AFA	00
3AF3	18	3AFB	00
3AF4	18	3AFC	00
3AF5	18	3AFD	00
3AF6	18	3AFE	00
3AF7	00	3AFF	FF

Address	Value
3B00	30
3B01	18
3B02	0C
3B03	00
3B04	00
3B05	00
3B06	00
3B07	00
3B08	00
3B09	00
3B0A	78
3B0B	0C
3B0C	7C
3B0D	CC
3B0E	76
3B0F	00
3B10	E0
3B11	60
3B12	7C
3B13	66
3B14	66
3B15	66
3B16	DC
3B17	00
3B18	00
3B19	00
3B1A	3C
3B1B	66
3B1C	60
3B1D	66
3B1E	3C
3B1F	00
3B20	1C
3B21	0C
3B22	7C
3B23	CC
3B24	CC
3B25	CC
3B26	76
3B27	00
3B28	00
3B29	00
3B2A	3C
3B2B	66
3B2C	7E
3B2D	60
3B2E	3C
3B2F	00
3B30	1C
3B31	36
3B32	30
3B33	78
3B34	30
3B35	30
3B36	78
3B37	00
3B38	00
3B39	00
3B3A	3E
3B3B	66
3B3C	66
3B3D	3E
3B3E	06
3B3F	7C
3B40	E0
3B41	60
3B42	6C
3B43	76
3B44	66
3B45	66
3B46	E6
3B47	00
3B48	18
3B49	00
3B4A	38
3B4B	18
3B4C	18
3B4D	18
3B4E	3C
3B4F	00
3B50	06
3B51	00
3B52	0E
3B53	06
3B54	06
3B55	66
3B56	66
3B57	3C
3B58	E0
3B59	60
3B5A	66
3B5B	6C
3B5C	78
3B5D	6C
3B5E	E6
3B5F	00

Address	Value
3B60	38
3B61	18
3B62	18
3B63	18
3B64	18
3B65	18
3B66	3C
3B67	00

Address	Value
3B70	00
3B71	00
3B72	DC
3B73	66
3B74	66
3B75	66
3B76	66
3B77	00

Address	Value
3B80	00
3B81	00
3B82	DC
3B83	66
3B84	66
3B85	7C
3B86	60
3B87	F0

Address	Value
3B90	00
3B91	00
3B92	DC
3B93	76
3B94	60
3B95	60
3B96	F0
3B97	00

Address	Value
3BA0	30
3BA1	30
3BA2	7C
3BA3	30
3BA4	30
3BA5	36
3BA6	1C
3BA7	00

Address	Value
3BB0	00
3BB1	00
3BB2	66
3BB3	66
3BB4	66
3BB5	3C
3BB6	18
3BB7	00

Address	Value
3B68	00
3B69	00
3B6A	6C
3B6B	FE
3B6C	D6
3B6D	D6
3B6E	C6
3B6F	00

Address	Value
3B78	00
3B79	00
3B7A	3C
3B7B	66
3B7C	66
3B7D	66
3B7E	3C
3B7F	00

Address	Value
3B88	00
3B89	00
3B8A	76
3B8B	CC
3B8C	CC
3B8D	7C
3B8E	0C
3B8F	1E

Address	Value
3B98	00
3B99	00
3B9A	3C
3B9B	60
3B9C	3C
3B9D	06
3B9E	7C
3B9F	00

Address	Value
3BA8	00
3BA9	00
3BAA	66
3BAB	66
3BAC	66
3BAD	66
3BAE	3E
3BAF	00

Address	Value
3BB8	00
3BB9	00
3BBA	C6
3BBB	D6
3BBC	D6
3BBD	FE
3BBE	6C
3BBF	00

Address	Data	Address	Data
3BC0	00	3BC8	00
3BC1	00	3BC9	00
3BC2	C6	3BCA	66
3BC3	6C	3BCB	66
3BC4	38	3BCC	66
3BC5	6C	3BCD	3E
3BC6	C6	3BCE	06
3BC7	00	3BCF	7C

Address	Data	Address	Data
3BD0	00	3BD8	0E
3BD1	00	3BD9	18
3BD2	7E	3BDA	18
3BD3	4C	3BDB	70
3BD4	18	3BDC	18
3BD5	32	3BDD	18
3BD6	7E	3BDE	0E
3BD7	00	3BDF	00

Address	Data	Address	Data
3BE0	18	3BE8	70
3BE1	18	3BE9	18
3BE2	18	3BEA	18
3BE3	18	3BEB	0E
3BE4	18	3BEC	18
3BE5	18	3BED	18
3BE6	18	3BEE	70
3BE7	00	3BEF	00

Address	Data	Address	Data
3BF0	76	3BF8	CC
3BF1	DC	3BF9	33
3BF2	00	3BFA	CC
3BF3	00	3BFB	33
3BF4	00	3BFC	CC
3BF5	00	3BFD	33
3BF6	00	3BFE	CC
3BF7	00	3BFF	33

Address	Data	Address	Data
3C00	00	3C08	F0
3C01	00	3C09	F0
3C02	00	3C0A	F0
3C03	00	3C0B	F0
3C04	00	3C0C	00
3C05	00	3C0D	00
3C06	00	3C0E	00
3C07	00	3C0F	00

Address	Data	Address	Data
3C10	0F	3C18	FF
3C11	0F	3C19	FF
3C12	0F	3C1A	FF
3C13	0F	3C1B	FF
3C14	00	3C1C	00
3C15	00	3C1D	00
3C16	00	3C1E	00
3C17	00	3C1F	00

Address	Value
3C20	00
3C21	00
3C22	00
3C23	00
3C24	F0
3C25	F0
3C26	F0
3C27	F0
3C28	F0
3C29	F0
3C2A	F0
3C2B	F0
3C2C	F0
3C2D	F0
3C2E	F0
3C2F	F0
3C30	0F
3C31	0F
3C32	0F
3C33	0F
3C34	F0
3C35	F0
3C36	F0
3C37	F0
3C38	FF
3C39	FF
3C3A	FF
3C3B	FF
3C3C	F0
3C3D	F0
3C3E	F0
3C3F	F0
3C40	00
3C41	00
3C42	00
3C43	00
3C44	0F
3C45	0F
3C46	0F
3C47	0F
3C48	F0
3C49	F0
3C4A	F0
3C4B	F0
3C4C	0F
3C4D	0F
3C4E	0F
3C4F	0F
3C50	0F
3C51	0F
3C52	0F
3C53	0F
3C54	0F
3C55	0F
3C56	0F
3C57	0F
3C58	FF
3C59	FF
3C5A	FF
3C5B	FF
3C5C	0F
3C5D	0F
3C5E	0F
3C5F	0F
3C60	00
3C61	00
3C62	00
3C63	00
3C64	FF
3C65	FF
3C66	FF
3C67	FF
3C68	F0
3C69	F0
3C6A	F0
3C6B	F0
3C6C	FF
3C6D	FF
3C6E	FF
3C6F	FF
3C70	0F
3C71	0F
3C72	0F
3C73	0F
3C74	FF
3C75	FF
3C76	FF
3C77	FF
3C78	FF
3C79	FF
3C7A	FF
3C7B	FF
3C7C	FF
3C7D	FF
3C7E	FF
3C7F	FF

Address	Value	Address	Value
3C80	00	3C88	18
3C81	00	3C89	18
3C82	00	3C8A	18
3C83	18	3C8B	18
3C84	18	3C8C	18
3C85	00	3C8D	00
3C86	00	3C8E	00
3C87	00	3C8F	00
3C90	00	3C98	18
3C91	00	3C99	18
3C92	00	3C9A	18
3C93	1F	3C9B	1F
3C94	1F	3C9C	0F
3C95	00	3C9D	00
3C96	00	3C9E	00
3C97	00	3C9F	00
3CA0	00	3CA8	18
3CA1	00	3CA9	18
3CA2	00	3CAA	18
3CA3	18	3CAB	18
3CA4	18	3CAC	18
3CA5	18	3CAD	18
3CA6	18	3CAE	18
3CA7	18	3CAF	18
3CB0	00	3CB8	18
3CB1	00	3CB9	18
3CB2	00	3CBA	18
3CB3	0F	3CBB	1F
3CB4	1F	3CBC	1F
3CB5	18	3CBD	18
3CB6	18	3CBE	18
3CB7	18	3CBF	18
3CC0	00	3CC8	18
3CC1	00	3CC9	18
3CC2	00	3CCA	18
3CC3	F8	3CCB	F8
3CC4	F8	3CCC	F0
3CC5	00	3CCD	00
3CC6	00	3CCE	00
3CC7	00	3CCF	00
3CD0	00	3CD8	18
3CD1	00	3CD9	18
3CD2	00	3CDA	18
3CD3	FF	3CDB	FF
3CD4	FF	3CDC	FF
3CD5	00	3CDD	00
3CD6	00	3CDE	00
3CD7	00	3CDF	00

Address	Value
3CE0	00
3CE1	00
3CE2	00
3CE3	F0
3CE4	F8
3CE5	18
3CE6	18
3CE7	18

Address	Value
3CE8	18
3CE9	18
3CEA	18
3CEB	F8
3CEC	F8
3CED	18
3CEE	18
3CEF	18

Address	Value
3CF0	00
3CF1	00
3CF2	00
3CF3	FF
3CF4	FF
3CF5	18
3CF6	18
3CF7	18

Address	Value
3CF8	18
3CF9	18
3CFA	18
3CFB	FF
3CFC	FF
3CFD	18
3CFE	18
3CFF	18

Address	Value
3D00	10
3D01	38
3D02	6C
3D03	C6
3D04	00
3D05	00
3D06	00
3D07	00

Address	Value
3D08	0C
3D09	18
3D0A	30
3D0B	00
3D0C	00
3D0D	00
3D0E	00
3D0F	00

Address	Value
3D10	66
3D11	66
3D12	00
3D13	00
3D14	00
3D15	00
3D16	00
3D17	00

Address	Value
3D18	3C
3D19	66
3D1A	60
3D1B	F8
3D1C	60
3D1D	66
3D1E	FE
3D1F	00

Address	Value
3D20	38
3D21	44
3D22	BA
3D23	A2
3D24	BA
3D25	44
3D26	38
3D27	00

Address	Value
3D28	7E
3D29	F4
3D2A	F4
3D2B	74
3D2C	34
3D2D	34
3D2E	34
3D2F	00

Address	Value
3D30	1E
3D31	30
3D32	38
3D33	6C
3D34	38
3D35	18
3D36	F0
3D37	00

Address	Value
3D38	18
3D39	18
3D3A	0C
3D3B	00
3D3C	00
3D3D	00
3D3E	00
3D3F	00

Address	Value
3D40	40
3D41	C0
3D42	44
3D43	4C
3D44	54
3D45	1E
3D46	04
3D47	00

Address	Value
3D48	40
3D49	C0
3D4A	4C
3D4B	52
3D4C	44
3D4D	08
3D4E	1E
3D4F	00

Address	Value
3D50	E0
3D51	10
3D52	62
3D53	16
3D54	EA
3D55	0F
3D56	02
3D57	00

Address	Value
3D58	00
3D59	18
3D5A	18
3D5B	7E
3D5C	18
3D5D	18
3D5E	7E
3D5F	00

Address	Value
3D60	18
3D61	18
3D62	00
3D63	7E
3D64	00
3D65	18
3D66	18
3D67	00

Address	Value
3D68	00
3D69	00
3D6A	00
3D6B	7E
3D6C	06
3D6D	06
3D6E	00
3D6F	00

Address	Value
3D70	18
3D71	00
3D72	18
3D73	30
3D74	66
3D75	66
3D76	3C
3D77	00

Address	Value
3D78	18
3D79	00
3D7A	18
3D7B	18
3D7C	18
3D7D	18
3D7E	18
3D7F	00

Address	Value
3D80	00
3D81	00
3D82	73
3D83	DE
3D84	CC
3D85	DE
3D86	73
3D87	00

Address	Value
3D88	7C
3D89	C6
3D8A	C6
3D8B	FC
3D8C	C6
3D8D	C6
3D8E	F8
3D8F	C0

Address	Value
3D90	00
3D91	66
3D92	66
3D93	3C
3D94	66
3D95	66
3D96	3C
3D97	00

Address	Value
3D98	3C
3D99	60
3D9A	60
3D9B	3C
3D9C	66
3D9D	66
3D9E	3C
3D9F	00

Address	Value
3DA0	00
3DA1	00
3DA2	1E
3DA3	30
3DA4	7C
3DA5	30
3DA6	1E
3DA7	00
3DA8	38
3DA9	6C
3DAA	C6
3DAB	FE
3DAC	C6
3DAD	6C
3DAE	38
3DAF	00
3DB0	00
3DB1	C0
3DB2	60
3DB3	30
3DB4	38
3DB5	6C
3DB6	C6
3DB7	00
3DB8	00
3DB9	00
3DBA	66
3DBB	66
3DBC	66
3DBD	7C
3DBE	60
3DBF	60
3DC0	00
3DC1	00
3DC2	00
3DC3	FE
3DC4	6C
3DC5	6C
3DC6	6C
3DC7	00
3DC8	00
3DC9	00
3DCA	00
3DCB	7E
3DCC	D8
3DCD	D8
3DCE	70
3DCF	00
3DD0	03
3DD1	06
3DD2	0C
3DD3	3C
3DD4	66
3DD5	3C
3DD6	60
3DD7	C0
3DD8	03
3DD9	06
3DDA	0C
3DDB	66
3DDC	66
3DDD	3C
3DDE	60
3DDF	C0
3DE0	00
3DE1	E6
3DE2	3C
3DE3	18
3DE4	38
3DE5	6C
3DE6	C7
3DE7	00
3DE8	00
3DE9	00
3DEA	66
3DEB	C3
3DEC	DB
3DED	DB
3DEE	7E
3DEF	00
3DF0	FE
3DF1	C6
3DF2	60
3DF3	30
3DF4	60
3DF5	C6
3DF6	FE
3DF7	00
3DF8	00
3DF9	7C
3DFA	C6
3DFB	C6
3DFC	C6
3DFD	6C
3DFE	EE
3DFF	00

Address	Byte	Address	Byte
3E00	18	3E08	18
3E01	30	3E09	0C
3E02	60	3E0A	06
3E03	C0	3E0B	03
3E04	80	3E0C	01
3E05	00	3E0D	00
3E06	00	3E0E	00
3E07	00	3E0F	00
3E10	00	3E18	00
3E11	00	3E19	00
3E12	00	3E1A	00
3E13	01	3E1B	80
3E14	03	3E1C	C0
3E15	06	3E1D	60
3E16	0C	3E1E	30
3E17	18	3E1F	18
3E20	18	3E28	18
3E21	3C	3E29	0C
3E22	66	3E2A	06
3E23	C3	3E2B	03
3E24	81	3E2C	03
3E25	00	3E2D	06
3E26	00	3E2E	0C
3E27	00	3E2F	18
3E30	00	3E38	18
3E31	00	3E39	30
3E32	00	3E3A	60
3E33	81	3E3B	C0
3E34	C3	3E3C	C0
3E35	66	3E3D	60
3E36	3C	3E3E	30
3E37	18	3E3F	18
3E40	18	3E48	18
3E41	30	3E49	0C
3E42	60	3E4A	06
3E43	C1	3E4B	83
3E44	83	3E4C	C1
3E45	06	3E4D	60
3E46	0C	3E4E	30
3E47	18	3E4F	18
3E50	18	3E58	C3
3E51	3C	3E59	E7
3E52	66	3E5A	7E
3E53	C3	3E5B	3C
3E54	C3	3E5C	3C
3E55	66	3E5D	7E
3E56	3C	3E5E	E7
3E57	18	3E5F	C3

Address	Data	Address	Data
3E60	03	3E68	C0
3E61	07	3E69	E0
3E62	0E	3E6A	70
3E63	1C	3E6B	38
3E64	38	3E6C	1C
3E65	70	3E6D	0E
3E66	E0	3E6E	07
3E67	C0	3E6F	03
3E70	CC	3E78	AA
3E71	CC	3E79	55
3E72	33	3E7A	AA
3E73	33	3E7B	55
3E74	CC	3E7C	AA
3E75	CC	3E7D	55
3E76	33	3E7E	AA
3E77	33	3E7F	55
3E80	FF	3E88	03
3E81	FF	3E89	03
3E82	00	3E8A	03
3E83	00	3E8B	03
3E84	00	3E8C	03
3E85	00	3E8D	03
3E86	00	3E8E	03
3E87	00	3E8F	03
3E90	00	3E98	C0
3E91	00	3E99	C0
3E92	00	3E9A	C0
3E93	00	3E9B	C0
3E94	00	3E9C	C0
3E95	00	3E9D	C0
3E96	FF	3E9E	C0
3E97	FF	3E9F	C0
3EA0	FF	3EA8	FF
3EA1	FE	3EA9	7F
3EA2	FC	3EAA	3F
3EA3	F8	3EAB	1F
3EA4	F0	3EAC	0F
3EA5	E0	3EAD	07
3EA6	C0	3EAE	03
3EA7	80	3EAF	01
3EB0	01	3EB8	80
3EB1	03	3EB9	C0
3EB2	07	3EBA	E0
3EB3	0F	3EBB	F0
3EB4	1F	3EBC	F8
3EB5	3F	3EBD	FC
3EB6	7F	3EBE	FE
3EB7	FF	3EBF	FF

Address	Value
3EC0	AA
3EC1	55
3EC2	AA
3EC3	55
3EC4	00
3EC5	00
3EC6	00
3EC7	00

Address	Value
3EC8	0A
3EC9	05
3ECA	0A
3ECB	05
3ECC	0A
3ECD	05
3ECE	0A
3ECF	05

Address	Value
3ED0	00
3ED1	00
3ED2	00
3ED3	00
3ED4	AA
3ED5	55
3ED6	AA
3ED7	55

Address	Value
3ED8	A0
3ED9	50
3EDA	A0
3EDB	50
3EDC	A0
3EDD	50
3EDE	A0
3EDF	50

Address	Value
3EE0	AA
3EE1	54
3EE2	A8
3EE3	50
3EE4	A0
3EE5	40
3EE6	80
3EE7	00

Address	Value
3EE8	AA
3EE9	55
3EEA	2A
3EEB	15
3EEC	0A
3EED	05
3EEE	02
3EEF	01

Address	Value
3EF0	01
3EF1	02
3EF2	05
3EF3	0A
3EF4	15
3EF5	2A
3EF6	55
3EF7	AA

Address	Value
3EF8	00
3EF9	80
3EFA	40
3EFB	A0
3EFC	50
3EFD	A8
3EFE	54
3EFF	AA

Address	Value
3F00	7E
3F01	FF
3F02	99
3F03	FF
3F04	BD
3F05	C3
3F06	FF
3F07	7E

Address	Value
3F08	7E
3F09	FF
3F0A	99
3F0B	FF
3F0C	C3
3F0D	BD
3F0E	FF
3F0F	7E

Address	Value
3F10	38
3F11	38
3F12	FE
3F13	FE
3F14	FE
3F15	10
3F16	38
3F17	00

Address	Value
3F18	10
3F19	38
3F1A	7C
3F1B	FE
3F1C	7C
3F1D	38
3F1E	10
3F1F	00

Address	Data
3F20	6C
3F21	FE
3F22	FE
3F23	FE
3F24	7C
3F25	38
3F26	10
3F27	00
3F28	10
3F29	38
3F2A	7C
3F2B	FE
3F2C	FE
3F2D	10
3F2E	38
3F2F	00
3F30	00
3F31	3C
3F32	66
3F33	C3
3F34	C3
3F35	66
3F36	3C
3F37	00
3F38	00
3F39	3C
3F3A	7E
3F3B	FF
3F3C	FF
3F3D	7E
3F3E	3C
3F3F	00
3F40	00
3F41	7E
3F42	66
3F43	66
3F44	66
3F45	66
3F46	7E
3F47	00
3F48	00
3F49	7E
3F4A	7E
3F4B	7E
3F4C	7E
3F4D	7E
3F4E	7E
3F4F	00
3F50	0F
3F51	07
3F52	0D
3F53	78
3F54	CC
3F55	CC
3F56	CC
3F57	78
3F58	3C
3F59	66
3F5A	66
3F5B	66
3F5C	3C
3F5D	18
3F5E	7E
3F5F	18
3F60	0C
3F61	0C
3F62	0C
3F63	0C
3F64	0C
3F65	3C
3F66	7C
3F67	38
3F68	18
3F69	1C
3F6A	1E
3F6B	1B
3F6C	18
3F6D	78
3F6E	F8
3F6F	70
3F70	99
3F71	5A
3F72	24
3F73	C3
3F74	C3
3F75	24
3F76	5A
3F77	99
3F78	10
3F79	38
3F7A	38
3F7B	38
3F7C	38
3F7D	38
3F7E	7C
3F7F	D6

Address	Value	Address	Value
3F80	18	3F88	18
3F81	3C	3F89	18
3F82	7E	3F8A	18
3F83	FF	3F8B	18
3F84	18	3F8C	FF
3F85	18	3F8D	7E
3F86	18	3F8E	3C
3F87	18	3F8F	18
3F90	10	3F98	08
3F91	30	3F99	0C
3F92	70	3F9A	0E
3F93	FF	3F9B	FF
3F94	FF	3F9C	FF
3F95	70	3F9D	0E
3F96	30	3F9E	0C
3F97	10	3F9F	08
3FA0	00	3FA8	00
3FA1	00	3FA9	00
3FA2	18	3FAA	FF
3FA3	3C	3FAB	FF
3FA4	7E	3FAC	7E
3FA5	FF	3FAD	3C
3FA6	FF	3FAE	18
3FA7	00	3FAF	00
3FB0	80	3FB8	02
3FB1	E0	3FB9	0E
3FB2	F8	3FBA	3E
3FB3	FE	3FBB	FE
3FB4	F8	3FBC	3E
3FB5	E0	3FBD	0E
3FB6	80	3FBE	02
3FB7	00	3FBF	00
3FC0	38	3FC8	38
3FC1	38	3FC9	38
3FC2	92	3FCA	10
3FC3	7C	3FCB	FE
3FC4	10	3FCC	10
3FC5	28	3FCD	28
3FC6	28	3FCE	44
3FC7	28	3FCF	82
3FD0	38	3FD8	38
3FD1	38	3FD9	38
3FD2	12	3FDA	90
3FD3	7C	3FDB	7C
3FD4	90	3FDC	12
3FD5	28	3FDD	28
3FD6	24	3FDE	48
3FD7	22	3FDF	88

Address	Value
3FE0	00
3FE1	3C
3FE2	18
3FE3	3C
3FE4	3C
3FE5	3C
3FE6	18
3FE7	00

Address	Value
3FE8	3C
3FE9	FF
3FEA	FF
3FEB	18
3FEC	0C
3FED	18
3FEE	30
3FEF	18

Address	Value
3FF0	18
3FF1	3C
3FF2	7E
3FF3	18
3FF4	18
3FF5	7E
3FF6	3C
3FF7	18

Address	Value
3FF8	00
3FF9	24
3FFA	66
3FFB	FF
3FFC	66
3FFD	24
3FFE	00
3FFF	00

3 BASIC

3.1 Der BASIC–Interpreter des CPC 464

Der CPC 464 hat einen schnellen und komfortablen BASIC–Interpreter, der in 16 KByte ROM untergebracht ist. Er belegt den Adreßbereich von &C000 bis &FFFF parallel zum Bildschirm–RAM. Für BASIC–Programm und –Variablen steht der Bereich von &0170 bis &AB80 zur Verfügung, das sind 43534 Byte.

Der Interpreter unterstützt fast alle Möglichkeiten, die von der Hardware und dem Betriebssystem her gegeben sind. Dazu gehören vor allem die Bildschirmausgabe mit bis zu 8 Fenstern, die hochauflösende Grafik, der Sound sowie die Eventverarbeitung. Damit ist es erstmals von BASIC aus möglich, mehrere 'Jobs' nebeneinander laufen zu lassen. Desweiteren bietet der BASIC–Interpreter eine Integerarithmetik mit 16–Bit–Zahlen (Wertebereich –32768 bis 32767) und eine Fließkommaarithmetik mit 8–Bit–Zweierexponent und 32–Bit–Mantisse, die eine Genauigkeit von 9 Dezimalstellen bei einem Wertebereich von ±1E–39 bis ±1E+38 garantiert.

Dabei sind Integer– und Fließkommaarithmetik nicht Teil des BASIC–Interpreters, sondern sind im Betriebssystem–ROM enthalten (Adresse &2E18 bis &37FF). Sie werden ebenso wie die übrigen Funktionen des Betriebssystems über die Sprungtabelle im oberen RAM–Bereich (&BB00 bis &BDF1) aufgerufen, die im Bedarfsfall geändert werden kann.

Auch sonst ermöglicht der BASIC–Interpreter ein komfortables Erstellen, Editieren und Ausführen von Programmen. Dem ersteren dient der Befehl AUTO, zum Editieren der EDIT–Befehl, der durch die Fähigkeiten des Betriebssystems kaum dem Arbeiten mit einem Bildschirmeditor nachsteht, sowie die Befehle RENUM, MERGE und DELETE. Auch bei der Ausführung des Programms braucht auf Komfort nicht verzichtet werden. Als Beispiel mögen die Fehlerbehandlung mit ON ERROR GOTO, die Typdefinition von Variablen mit DEFtyp, das selektive Löschen von Feldern mit ERASE, die Ein– und Ausgabe von Zahlen als Dezimal–, Binär– oder Hexzahl, selbstdefinierte Funktionen mit mehreren Argumenten und allen Datentypen und Programmstrukturen wie IF...THEN...ELSE, FOR...NEXT und WHILE...WEND dienen. Die Neubelegung von Tasten und Funktionstasten ist ebenso einfach möglich wie die Definition eigener Sonderzeichen auf dem Bildschirm. Ein TRACE–Befehl fehlt ebensowenig wie ein umfassendes PRINT USING.

Nach dieser kurzen Übersicht soll auf die Eingabe und Ablage von BASIC–Zeilen sowie die Ausführung durch den BASIC–Interpreter etwas nä-

her eingegangen werden. Mit diesen Kenntnissen können Sie nicht nur 'das Letzte' aus dem BASIC–Interpreter herausholen, sie sind auch die Grundlage zum Schreiben eigener BASIC–Erweiterungen, von denen später noch einige Beispiele folgen.

Die Eingabe von BASIC–Zeilen

Wenn Sie eine BASIC–Zeile eingeben, so wird sie zuerst in einen 256 Byte großen Puffer übernommen, der von Adresse &ACA4 bis &ADA3 liegt. Dort steht die Eingabe im Klartext. Beginnt die Zeile mit einer Zeilennummer, so wird diese in eine 16–Bit–Binärzahl ungewandelt und in einen zweiten Puffer für die umgewandelte Zeile abgelegt. Dieser Puffer ist 300 Zeichen groß und liegt vor dem BASIC–Programm von Adresse &40 bis &16F. Dann wird die Eingabezeile auf BASIC–Schlüsselworte durchsucht. Diese Schlüsselworte werden durch ein Byte ersetzt, das sogenannte Token. So wird z.B aus 'AFTER' das Token &80. Die Tokens sämtlicher Befehlsworte und der BASIC–Operatoren wie '=' und 'AND' haben Werte größer als 127, das 7. Bit ist also gesetzt. BASIC–Funktionen wie EXP oder ROUND haben Tokens zwischen 0 und &7F. Um sie von normalen ASCII–Zeichen zu unterscheiden, werden sie durch ein vorangestelltes &FF gekennzeichnet. Der Doppelpunkt zum Trennen zweier Statements wird durch den Kode &01 dargestellt, das Zeilenende wird durch &00 abgeschlossen. Konnte eine Buchstabenfolge nicht als Befehl oder Funktion identifiziert werden, so wird sie als Variablenname behandelt. Eine Variablenname kann bis zu 40 Zeichen lang sein, die alle signifikant sind. Zwischen Klein– und Großschreibung wird dabei nicht unterschieden. Nehmen wir an, wir hätten folgende Zeile eingegeben:

10 start=77

Nach der Zeilennummer wird jetzt folgendes abgelegt:

&0D &00 &00 &73 &74 &61 &72 &F4 &EF &19 &4D &00

Dabei bedeutet die &0D, daß es sich um eine Variable ohne Typkennzeichen handelt. Dann folgen zwei Nullbytes, auf die wir später noch zurück kommen. Nun folgt der Name der Variablen, die ASCII–Codes für s,t,a und r. Beim letzten Zeichen 't' wird zum ASCII–Code &74 noch &80 dazu addiert (das oberste Bit wird gesetzt) und wir erhalten &F4. Der Code &EF ist das Token für '='. Das nachfolgende &19 kündigt eine Ein–Byte–Konstante an: &4D ist gleich 77. Die abschließende Null bedeutet das Ende der Zeile.

Vor der Zeilennummer folgen noch zwei Bytes, die die Länge der Zeile angeben:

&12 &00 &0A &00

Die Zeile ist also &12 + 256 * &00 gleich 18 Bytes lang und hat die Zeilennummer &0A + 256 * &00 gleich 10.

Sie sehen also, daß im Gegensatz zu anderen BASIC–Interpretern Konstanten im Programmtext nicht als ASCII–Texte abgelegt, sondern bereits in die Binärform überführt worden sind. Dies hat einen ganz entscheidenen Vorteil. Die zeitintensive Umwandlung vom ASCII ins Binärformat braucht nur einmal, nämlich bei der Eingabe der Zeile zu geschehen und nicht bei jedem Ausführen der Zeile noch einmal. Dies bedeutet einen nicht unerheblichen Geschwindigkeitsvorteil beim Ausführen der Programme.

Der CPC 464 kennt noch eine ganze Reihe von numerischen Konstanten, die durch ein entsprechendes Token gekennzeichnet sind. Konstanten, die nur aus einer Ziffer bestehen, also die Zahlen von 0 bis 9, werden durch die Token &0E bis &17 kodiert, belegen also nur ein Byte im Programmtext. Das Token &19 für Ein–Byte–Werte haben wir bereits kenngelernt. Für Zwei–Byte–Integerwerte gibt es drei verschiedene Token, je nachdem, ob die Konstante dezimal, binär oder hex eingegeben wurde. Die Ablage mit Lo– und Hi–Byte ist in allen drei Fällen gleich.

&1A Zwei-Byte-Wert, dezimal
&1B Zwei-Byte-Wert, binär
&1C Zwei-Byte-Wert, hexadezimal

Handelt es sich um keine ganze Zahl oder ist der Betrag größer als 32767, so wird sie als Fließkommawert abgelegt, der durch das Token &1F gekennzeichnet wird. Dann folgen 5 Bytes, der Fließkommawert. Auf die Fließkommazahlen wird an anderer Stelle noch eingegangen.

Eine Sonderstellung nehmen in diesem Zusammenhang die Zeilennummern ein, wie sie z.B. nach Befehlen wie GOTO, GOSUB oder RUN folgen. Sie werden ebenfalls als 16–Bit–Binärzahl abgelegt, jedoch durch das Token &1E gekennzeichnet.

Wird ein Programm ausgeführt und trifft es z.B. auf einen GOTO–Befehl, so liest es die Zeilennummer und muß das ganze Programm nach dieser Zeile durchsuchen. Besonders bei größeren Programmen kann das relativ lange dauern. Oft werden GOTO– und GOSUB–Befehle in Programmschleifen benutzt und hundert oder tausende Male durchlaufen. Dabei können die Suchzeiten dann einen Großteil der Programmausführungszeit ausmachen. Der BASIC–Interpreter des CPC 464 führt diese Zeilensuche nur einmal durch. Hat er die Zeile gefunden, so ersetzt er die Zeilennummer hinter dem GOTO–Befehl durch die Adresse der Zeile, die er gefunden hat. Damit er die Adresse von einer Zeilennummer unterschei-

den kann, ändert er das Token &1E in &1D, das Token für die Zeilenadresse. Wird der GOTO-Befehl dann nochmal durchgeführt, hat der Interpreter direkt die Adresse, an die die Programmausführung verzweigen kann, was viel Zeit spart.

Dieses Verfahren bringt jedoch bei Befehlen, die die Zeilennummer als solche brauchen, Schwierigkeiten. Wenn der LIST-Befehl die Zeile ausgeben soll, muß er ja die Zeilennummer ausgeben und nicht deren Adresse. Dieses Problem ist jedoch leicht gelöst. Wenn die Zeilenadresse bekannt ist, kann einfach von dort die Zeilennummer geholt werden, da ja, wie wir oben gesehen haben, die Zeilennummer in der Zeile abgelegt ist. Wenn Zeilen gelöscht oder eingefügt werden, muß eine Ersetzung der Zeilennadresse durch die Zeilennumern erfolgen, da die Adressen sich danach ja ändern. Dies betrifft jedoch nur die Ein- und Ausgabe von Programmzeilen und wird durch die bedeutend schnellere Programmausführung bei weitem wieder ausgeglichen.

Die Programmausführung durch den BASIC-Interpreter

Die Ausführung eines Statements durch den BASIC-Interpreter läßt sich vereinfacht folgendermaßen darstellen. Jede Programmzeile beginnt wie beschrieben mit der Programmlänge und der Zeilennummer. Danach kommt der eigentliche BASIC-Befehl. Der Interpreter prüft nun, ob es sich um ein Befehlstoken handelt, das durch einen Wert zwischen &80 und &DC gekennzeichnet ist. Ist dies der Fall, so benutzt er dieses Token als Zeiger in eine Tabelle, die die Adressen sämtlicher BASIC-Befehle enthält. Der BASIC-Befehl wird als Unterprogramm ausgeführt. Danach wird wieder in die sogenannte Interpreterschleife zurückgekehrt. Begann die Anweisung jedoch nicht mit einem Befehlstoken, so wird zum LET-Befehl verzweigt.

Der wohl wichtigste Teil des BASIC-Interpreters ist die Ausdrucksberechnung. Der CPC 464 unterscheidet dabei zwischen drei Typen von Ausdrücken: Integer, Fließkomma und String. Wenn z.B. eine Wertzuweisung an eine Variable ausgeführt wird oder wenn ein Parameter zu einem Befehl berechnet werden soll, so wird eine Routine aufgerufen, die den Ausdruck berechnet und den Wert sowie den Typ des Ausdrucks bereitstellt. Der Variablentyp kann drei Werte annehmen:

2 Integer
3 String
5 Fließkomma

Diese Typnummer ist gleichzeitig die Länge der Variablen. Bei einem String ist das der sogenannte Descriptor, der Länge und Adresse enthält (siehe auch Kapitel über den Variablenpointer). Stimmen nun Type eines

Ausdrucks und einer Variablen, an die dieser Ausdruck zugewiesen werden soll, nicht überein, so wird versucht, eine Typumwandlung durchzuführen. Das ist jedoch nur bei den numerischen Typen Integer und Fließkomma möglich. Diese Umwandlung kostet natürlich Rechenzeit. Deshalb sollte man immmer dort, wo es möglich ist, Integervariablen einsetzen. Die Praxis hat gezeigt, daß oft in 90% der Fälle Integervariablen eingesetzt werden können. Dadurch entfällt nicht nur eine Typumwandlung, sondern die Integerarithmetik ist auch noch bedeutend schneller als die Fließkommaarithmetik. Ganz besonders sollte dies für die Laufvariablen in FOR–NEXT–Schleifen und sonstige Zähler gelten.

Wird dagegen versucht, einen Stringausdruck einer numerischen Variablen oder umgekehrt zuzuweisen, so wird die Fehlermeldung 'Type mismatch' ausgegeben. Eine Umwandlung von String nach numerisch und umgekehrt ist nur explizit mit den Funktionen VAL und STR$ möglich.

3.2 Der BASIC-Stack

Ein Stack oder Stapelspeicher dient zum Ablegen von Daten nach dem 'Last in - First out' Prinzip. Der Prozessor benutzt dazu den Speicherbereich ab &C000. Vor jedem Eintrag wird der Stackpointer (Stapelzeiger) dekrementiert. Werden Daten wieder vom Stack geholt, so wird anschließend der Stapelzeiger wieder inkrementiert. Der Prozessorstack dient z.B. zur Ablage von Rücksprungadressen bei Unterprogrammaufrufen und erlaubt durch das Zugriffsprinzip eine Schachtelung von Unterprogrammen.

Der BASIC-Interpreter braucht zur Ablage der Parameter von GOSUB-Aufrufen, FOR-NEXT- und WHILE-WEND-Schleifen ebenfalls einen Stack, der erst eine Schachtelung dieser Programmstrukturen möglich macht. Dazu wird nun nicht der Prozessorstack mitbenutzt, sondern es existiert ein eigener BASIC-Stack, der 512 Bytes groß ist und bei Adresse &AE8B beginnt. Im Gegensatz zum Prozessorstack wächst dieser Stack mit zunehmenden Einträgen zu höheren Adressen hin bis maximal &B08A. Als Stackpointer dient die Speicherstelle &B08B/&B08C.

Betrachten wir zuerst, welche Parameter bei einem GOSUB-Befehl auf dem Stack abgelegt werden.

&00/&01 — **Kennzeichen der GOSUB-Klasse**

Lo
Hi — **Adresse der Anweisung nach dem GOSUB-Befehl**

Lo
Hi — **Zeilenadresse der GOSUB-Anweisung**

&06 — **Größe des Stackeintrags**

Zuerst wird also ein Byte abgelegt, was den Typ der GOSUB-Anweisung bestimmt. Bei einem normalen GOSUB-Befehl ist dies ein Nullbyte. Handelt es sich jedoch um den Aufruf eines Unterprogramms, die aus einem AFTER oder EVERY-Befehl herrührt, wird zur Unterscheidung eine Eins auf dem Stack abgelegt. Dann folgen die Adresse des nächsten Befehls nach dem GOSUB-Befehl sowie die Adresse der Zeile, in der der GOSUB-Befehl steht. Damit der Stackeintrag beim RETURN-Befehl wieder identifiziert werden kann, wird noch ein Byte auf dem Stack abgelegt, das die Länge des Stackeintrags angibt und damit implizit einen GOSUB-Datensatz kennzeichnet.

Die Daten einer WHILE-WEND-Schleife werden in ähnlicher Form abgelegt.

Lo	**Zeilenadresse des**
Hi	**WHILE-Befehls**
Lo	**Adresse des**
Hi	**WEND-Befehls**
Lo	**Adresse der**
Hi	**WHILE-Bedingung**
&07	**Größe des Stackeintrags**

Der Eintrag enthält also 3 Adressen und als Kennbyte eine 7, die Anzahl der Stackeinträge.

Bei der FOR-NEXT-Schleife wird es etwas komplizierter. Hierbei wird unterschieden, ob die Laufvariable vom Typ Integer oder Real ist. Im ersteren Fall ist nicht nur die Ausführungszeit kürzer, sondern der Stackbedarf ist ebenfalls kleiner. Betrachten wir zuerst den Aufbau bei einer Integerschleife.

Lo	**Adresse der**
Hi	**Laufvariablen**
Lo	**Endwert der**
Hi	**Laufvariablen**
Lo	**STEP-Wert**
Hi	
Sgn	**Vorzeichen des STEP-Werts**
Lo	**Adresse der**
Hi	**FOR-Anweisung**
Lo	**Zeilenadresse der**
Hi	**FOR-Anweisung**
Lo	**Adresse der**
Hi	**NEXT-Anweisung**
Lo	**Zeilenadresse**

Hi	der NEXT-Anweisung
&10	Größe des Stackeintrags

Der Stackeintrag für eine FOR-NEXT-Scheife mit Integervariable ist also 16 Bytes groß. Wird die Schleife mit einer Real-Variablen durchlaufen, so werden 22 Bytes im Stack belegt.

Lo Hi	Adresse der Laufvariablen
5 Bytes Fließ- kommawert	Endwert der Laufvariablen
5 Bytes Fließ- kommawert	STEP-Wert
Sgn	Vorzeichen des STEP-Werts
Lo Hi	Adresse der FOR-Anweisung
Lo Hi	Zeilenadresse der FOR-Anweisung
Lo Hi	Adresse der NEXT-Anweisung
Lo Hi	Zeilenadresse der NEXT-Anweisung
&16	Größe des Stackeintrags

Außer zur Abspeicherung von Schleifenstrukturen wird der BASIC-Stack noch zur Speicherung von Zwischenausdrücken bei numerischen Berechnungen benutzt, z.B. bei verschachtelten Klammerausdrücken und zur Realisierung einer Hierarchie bei den arithmetischen und logischen Operatoren.

3.3 Die BASIC-Vektoren

Wollen Sie eigene BASIC-Befehle oder Funktionen realisieren, so gibt es außer der Anwendung von Erweiterungsroms oder RSX noch eine weitere Möglichkeit zur Implementierung. Dies geschieht über die sogenannten BASIC-Vektoren.

Der BASIC-Interpreter ruft dazu an allen strategisch wichtigen Stellen ein Unterprogramm im RAM auf, das normalerweise nur aus einem RET-Befehl besteht und so keinen Einfluß hat. Soll z.B. ein Befehl ausgeführt werden, so wird geprüft, ob das Statement mit einem gültigen Befehlstoken beginnt. Ist dies der Fall, so wird aus dem Token die Adresse des zugehörigen Befehls berechnet und dorthin verzweigt. Wurde hingegen kein gültiger Befehl erkannt, so wird ein SYNTAX ERROR gemeldet. Vor der Ausgabe der Fehlermeldung - und dies ist der Trick dabei - wird jedoch zuerst die oben beschriebene Routine im RAM aufgerufen, die normalerweise nur aus einem RET-Befehl besteht und demzufolge direkt zurückkehrt. Wollen Sie einen eigenen Befehl 'einbauen', so brauchen Sie nur den entsprechenden RET-Befehl im RAM durch einen Sprung auf Ihre eigene Routine zu ersetzen, die auf einen neuen gültigen Befehl prüft und ihn ausführt.

Diese Methode ist nicht nur flexibler als die RSX-Methode - man ist nicht auf Integerargumente beschränkt (siehe Kapitel 3.5.2) - sondern ist auch in der Ausführung schneller, da nicht die Tabelle sämtlicher Erweiterungen nach dem richtigen Befehlswort durchsucht werden muß.

Die folgende Tabelle enthält die Adressen der neun Routinen, die vom Benutzer manipuliert werden können. Dabei wird eine Routine (wohl versehentlich) vom BASIC-Interpreter nicht angesprungen, sondern direkt zur Fehlerausgabe verzweigt (siehe BASIC-Listing, Adresse &D078).

AC01 - AC03	Patch für READY-Modus
AC04 - AC06	Patch für ERROR-Einsprung
AC07 - AC09	Patch für Befehl ausführen
AC0A - AC0C	Patch für Funktionsberechnung
AC0D - AC0F	Patch für Konstante holen (nicht benutzt)
AC10 - AC12	Patch für Eingabe, Zeile in Token wandeln
AC13 - AC15	Patch für Ausgabe, Token listen
AC16 - AC18	Patch für Eingabe, Umwandlung von Ziffern
AC19 - AC1B	Patch für Operatoren

Das folgende Beispiel soll eine Anwendung dieser Vektoren demonstrieren. Dazu wird der schon als Token vorhandene Befehl SWAP zum Austausch zweier Variabler benutzt.

```
                    ;Variablen-Swap
                    ;LE 15/12/84

00E7                SWAP     EQU   &E7 ; SWAP-Token
CA94                ERROR    EQU   &CA94 ; Fehler-Aussprung
000D                MISM     EQU   13 ; 'TYPE MISMATCH'
D686                GETVAR   EQU   &D686 ; Variable suchen
DD37                CHECK    EQU   &DD37 ; Zeichen prüfen
DD3F                BLANK    EQU   &DD3F ; Blanks überlesen

AC07                         ORG   &AC07

AC07  C30080                 JP    SWPNEW ; Vektor verbiegen

8000                         ORG   &8000

8000  FECE          SWPNEW   CP    SWAP*2 & &FF ; SWAP-Token
?
8002  C0                     RET   NZ ; 'SYNTAX ERROR'

8003  D1                     POP   DE ; Rücksprungadresse vom
Stack
8004  CD3FDD                 CALL  BLANK
8007  CD86D6                 CALL  GETVAR ; Variable suchen
800A  CD37DD                 CALL  CHECK
800D  2C                     DEFB  "," ; auf Komma prüfen
800E  ED534C80               LD    (VAR1),DE
8012  C5                     PUSH  BC ; und Typ merken
8013  CD86D6                 CALL  GETVAR
8016  ED534E80               LD    (VAR2),DE
801A  79                     LD    A,C
801B  C1                     POP   BC
801C  B9                     CP    C ; Typen vergleichen
801D  2805                   JR    Z,TYPOK

801F  1E0D                   LD    E,MISM
8021  C394CA                 JP    ERROR

8024  E5            TYPOK    PUSH  HL ; Programmzeiger merken
8025  0600                   LD    B,0
8027  2A4E80                 LD    HL,(VAR2)
802A  114780                 LD    DE,TEMP
802D  C5                     PUSH  BC
802E  EDB0                   LDIR  ; Variable 2 nach TEMP
8030  C1                     POP   BC
8031  2A4C80                 LD    HL,(VAR1)
8034  ED5B4E80               LD    DE,(VAR2)
```

```
8038  C5                  PUSH  BC
8039  EDB0                LDIR  ; Variable 1 nach Variable 2
803B  C1                  POP   BC
803C  214780              LD    HL,TEMP
803F  ED5B4C80            LD    DE,(VAR1)
8043  EDB0                LDIR  ; TEMP nach Variable 1
8045  E1                  POP   HL ; Programmzeiger zurück
8046  C9                  RET

8047            TEMP      DEFS  5 ; Zwischenspeicher
804C            VAR1      DEFS  2 ; Adresse Variable 1
804E            VAR2      DEFS  2 ; Adresse Variable 2
```

In diesem Zusammenhang ist noch die Speicherstelle &AC00 von Bedeutung. Normalerweise steht dort eine Null. Dies bewirkt, daß eine BASIC-Zeile so übernommen wird, wie sie eingegeben wurde. Wird diese Speicherstelle jedoch mit einem Wert ungleich Null geladen, so werden überflüssige Leerzeichen in der Eingabezeile ignoriert und nicht mit abgespeichert.

Wenn Sie z.B. folgende Zeile eingeben,

```
10 FOR i=1 TO 100:          PRINT     "Hallo"     :     NEXT
```

so wird daraus

10 FOR i=1 TO 100:PRINT"Hallo":NEXT   Dies kann z.B. dann nützlich sein, wenn der Speicherplatz knapp ist. Das Programm wird dadurch soweit wie möglich komprimiert. Der Nachteil dieser Methode ist jedoch, daß das Programm an übersichtlichkeit verlieren kann. Programmstrukturen können so schlechter sichtbar gemacht werden.

```
10 FOR I=1 TO 100
20        PRINT "Hallo"
30 NEXT
```

3.4 Das BASIC-RAM

In der folgenden Aufstellung finden Sie die Bedeutung der vom BASIC-Interpreter benutzen RAM-Adressen.

AB80 - ABFF	Zeichenmatrix für CHR$(240) bis CHR$(255)
AC00	Flag für zusätzliche Blanks ignorieren
AC01 - AC03	Patch für READY-Modus
AC04 - AC06	Patch für ERROR-Einsprung
AC07 - AC09	Patch für Befehl ausführen
AC0A - AC0C	Patch für Funktionsberechnung
AC0D - AC0F	Patch für Konstante holen (nicht benutzt)
AC10 - AC12	Patch für Eingabe, Zeile in Token wandeln
AC13 - AC15	Patch für Ausgabe, Token listen
AC16 - AC18	Patch für Eingabe, Umwandlung von Ziffern
AC19 - AC1B	Patch für Operatoren
AC1C	Flag für AUTO-Modus
AC1D - AC1E	Zeilennummer für AUTO
AC1F - AC20	Inkrement für AUTO
AC21	aktuelle Streamnummer
AC22	Eingabekanal
AC23	aktuelle PRINTER-Position
AC24	WIDTH
AC25	aktuelle Position auf Kassette
AC26	Flag für ersten FOR-NEXT-Durchlauf
AC27 - AC2B	Zwischenspeicher für FOR-Variable
AC2C - AC2D	Adresse des zugehörigen NEXT-Befehls
AC2E - AC2F	Adresse des zugehörigen WEND-Befehls
AC34 - AC35	ON BREAK Adresse
AC36 - AC37	
AC38 - AC43	Sound-Queue 0
AC44 - AC4F	Sound-Queue 1
AC50 - AC5B	Sound-Queue 2
AC5C - AC6D	Event-Block 0
AC6E - AC7F	Event-Block 1
AC80 - AC91	Event-Block 2
AC92 - ACA3	Event-Block 3
ACA4 - ADA3	Eingabepuffer
ADA6 - ADA7	Adresse der ERROR-Zeile
ADA8 - ADA8	Programmzeiger nach ERROR
ADAA	ERROR-Nummer
ADAB - ADAC	Programmzeiger nach Unterbrechung
ADAD - ADAE	Zeilenadresse nach Unterbrechung
ADAF - ADB0	Adresse der ON-ERROR-Routine
ADB1	Fehlerbehandlungsroutine aktiv

ADB2	Sound-Parameter Kanal-Status
ADB3	Sound-Parameter Lautstärken-Hüllkurve
ADB4	Sound-Parameter Ton-Hüllkurve
ADB5 - ADB6	Sound-Parameter Periode
ADB7	Sound-Parameter Geräusch-Periode
ADB8	Sound-Parameter Lautstärke
ADB9 - ADBA	Sound-Parameter Dauer
ADBB - ADBC	ENV & ENV
ADCB - ADCF	Zwischenspeicher für Fließkommazahl
ADD0 - AE03	Tabelle für skalare Variable
AE04 - AE05	Tabelle für FN
AE06 - AE0B	Tabelle für Arrays
AE0C - AE25	Variablentypen A-Z
AE2D	Trennzeichen bei INPUT-Befehl
AE2E - AE2F	Zeilenadresse bei READ-Befehl
AE30 - AE31	DATA-Zeiger
AE32 - AE33	Speicher für BASIC-Stackpointer
AE34 - AE35	Adresse des aktuellen Statements
AE36 - AE37	Adresse der aktuellen Programmzeile
AE38	TRACE-Flag
AE3F - AE40	Startadresse bei LOAD
AE41	Flag für CHAIN MERGE
AE42	Filetyp
AE43 - AE44	Filelänge
AE45	Flag für geschützes Programm
AE46 - AE78	Puffer für Umwandlung nach ASCII
AE72 - AE73	Adresse für CALL-Befehl
AE74	Konfiguration für CALL-Befehl
AE75 - AE76	hl während CALL-Befehl
AE77 - AE78	sp während CALL-Befehl
AE79	Tabulatorweite
AE7B - AE7C	Zeiger auf HIMEM
AE7D - AE7E	Zeiger auf Ende freies RAM
AE7F - AE80	Zeiger auf Start freies RAM
AE81 - AE82	Zeiger auf Programmstart
AE83 - AE84	Zeiger auf Programmende
AE85 - AE86	Zeiger auf Variablenstart
AE87 - AE88	Zeiger auf Arraystart
AE89 - AE8A	Zeiger auf Arrayende
AE8B - B08A	BASIC-Stack
B08B - B08C	BASIC-Stackpointer
B08D - B08E	Beginn der Strings
B08F - B090	Ende der Strings
B09A - B09B	Zeiger in Stringdescriptorstack
B09C - B0B9	Stringdescriptorstack
B0BA - B0BC	Stringdescriptor
B0C1	Variablentyp

B0C2 - B0C3	Integervariable bzw.
	Adresse einer Fließkommazahl
	Zeiger auf Stringdescriptor
B8E4 - B8E7	RND-Wert
B8E8 - B8EC	Zwischenspeicher für Fließkommazahl
B8ED - B8F1	Zwischenspeicher für Fließkommazahl
B8F2 - B8F6	Zwischenspeicher für Fließkommazahl
B8F7	Flag für DEG/RAD

3.5 BASIC und Maschinensprache

3.5.1 Der CALL-Befehl

Das Bindeglied zwischen BASIC und Maschinensprache ist der CALL-Befehl. Mit ihm ist es möglich, aus einem BASIC-Programm heraus ein Maschinenprogramm aufzurufen. Zum CALL-Befehl gehört noch eine 16-Bit-Adresse, die besagt, an welcher Stelle das Maschinenprogramm steht, z.B.

CALL &8000

Damit wird ein Maschinenprogramm ab der Adresse &8000 oder dezimal 32768 aufgerufen. Wird das Maschinenprogramm mit dem RET-Befehl beendet, wird die Kontrolle wieder zurück an den Interpreter gegeben, der in der Ausführung des BASIC-Programms fortfährt.

Beim CALL-Befehl sind Betriebssystem und BASIC-Interpreter nicht direkt zugänglich; im gesamten 64K-Adreßbereich ist RAM selektiert. Selbstverständlich lassen sich jedoch Betriebssystemroutinen über die Einsprungadressen im Bereich ab &B000 aufrufen. Diese Routinen sorgen selbsttätig für die erforderliche ROM/RAM-Konfiguration. Wollen Sie während eines CALL-Befehls auf Routinen des BASIC-Interpreters oder Betriebssystemroutinen zugreifen, die nicht über Vektoren angesprungen werden, so können Sie dazu die RST 3 und RST 5 Routinen benutzen, die die Umschaltung realisieren.

Der CALL-Befehl erlaubt es aber weiterhin noch Parameter von BASIC aus an die Routine zu übergeben. Dazu können hinter der Adresse durch Komma getrennt bis zu 32 Parameter übergeben werden, die der Maschinenroutine dann zur Verfügung stehen. Diese Parameter müssen wie die Adresse selbst einen 16-Bit-Wert ergeben. Sie werden von BASIC auf dem Stack abgelegt. Der BASIC-Interpreter übergibt die Basisadresse dieses Parameterblocks im IX-Register. Im Akku steht zusätzlich noch, wieviele Parameter übergeben wurden. Der letzte Parameter steht also an Adresse IX, der vorletzte an Adresse IX+2 und der erste Parameter an Adresse IX + 2*(A-1).

Während des CALL-Befehls können sämtliche Registerinhalte verändert werden (bezüglich der Benutzung der Zweitregister siehe im Kapitel Firmware). Auch der Stackpointer darf verändert werden, sofern sichergestellt ist, daß beim abschließenden RET die richtige Rücksprungadresse vom Stack geholt wird.

Bei der Benutzung des CALL-Befehls sind Ihrer Phantasie keine Grenzen gesetzt. Sie können z.B. erweiterte Grafikfunktionen zur Verfügung stel-

len wie Kreise zeichnen, Fläche ausfüllen usw.

Die Übergabe von Parametern von der Maschinenroutine zurück ans BASIC ist nicht vorgesehen, ist jedoch über einen Umweg möglich. Soll z.B. das Ergebnis eines Maschinenprogramms einer Variablen zugewiesen werden, so kann dem CALL-Befehl deren Adresse übergeben werden. Dies ist mit dem 'Klammeraffen' möglich.

CALL &AB00,@A

Damit steht dem Maschinenprogramm die Adresse der Variablen A zur Verfügung und es kann direkt der Wert der Variablen verändert werden. Diese Möglichkeit ist im Kapitel über den Variablenpointer näher beschrieben.

3.5.2 BASIC-Erweiterungen mit RSX

Betriebssystem und BASIC des CPC 464 unterstützen eine Möglichkeit, eigene Befehle ins BASIC einzubinden, die man RSX nennt. Das ist eine Abkürzung für 'Resident System eXtension'. Diese Erweiterungen können von BASIC aus durch einen Namen aufgerufen werden und erlauben eine Parameterübergabe, wie sie beim CALL-Befehl schon beschrieben wurde. Wenn wir z.B eine Grafikerweiterung schreiben wollen, die uns ein Quadrat auf den Bildschirm zeichnet, so kann der Aufruf dazu so aussehen:

10 **|QUADRAT,100,100,50**

Damit soll ein Quadrat gezeichnet werden, dessen linke obere Ecke die Koordinaten 100,100 hat mit einer Kantenlänge von 50 Punkten.

Wie Sie sehen, wird eine Befehlserweiterung durch einen vorangestellten senkrechten Strich (Shift @) gekennzeichnet.

Eine derartige Befehlserweiterung kann in einem Extensionrom stecken, wie es der Fall ist, wenn Sie ein Diskettenlaufwerk angeschlossen haben oder aber auch im RAM. Damit haben wir die Möglichkeit, eigene Erweiterungen zu schreiben. Damit das Betriebssystem weiß, wo es nach einer solchen Erweiterung suchen soll, muß die Erweiterung erst 'eingebunden' werden. Dazu dient eine Routine des Betriebssystems: KL LOG EXT. Das folgende Beispiel realisiert den oben angesprochenen Befehl zum Zeichnen eines Quadrat und demonstriert, wie die Einbindung vonstatten geht.

```
; RSX-BEFEHLSERWEITERUNG
```

```
                    ; LE 21/12/84

BCD1                LOGEXT   EQU   &BCD1 ; Erweiterung einbin-
den
BBC6                ASKCUR   EQU   &BBC6 ; Grafik-Cursor holen
BBC0                MOVABS   EQU   &BBC0 ; Grafik-Cursor setzen
BBF9                DRAWRE   EQU   &BBF9 ; Linie relativ ziehen
BDC7                CHGSGN   EQU   &BDC7 ; Vorzeichen ändern

8000                         ORG   &8000

8000  010980                 LD    BC,RSX ; Adresse der RSX-Be-
fehlstabelle
8003  211680                 LD    HL,KERNAL ; 4 Byte RAM für
Kernal
8006  C3D1BC                 JP    LOGEXT ; Erweiterung einbin-
den

8009  0E80          RSX      DEFW  TABLE ; Adresse der Befehls-
worte
800B  C31A80                 JP    QUADRAT

800E  51554144      TABLE    DEFM  "QUADRA"
8014  D4                     DEFB  "T"+&80
8015  00                     DEFB  0 ; Ende der Tabelle

8016                KERNAL   DEFS  4 ; Speicher für Kernal

801A  FE03          QUADRA   CP    3 ; drei Parameter ?
801C  C0                     RET   NZ
801D  CDC6BB                 CALL  ASKCURS ; Grafik-Cursor ho-
len
8020  D5                     PUSH  DE ; X-Koordinate merken
8021  E5                     PUSH  HL ; Y-Koordinate merken
8022  DD5605                 LD    D,(IX+5)
8025  DD5E04                 LD    E,(IX+4) ; X-Koordinate
8028  DD6603                 LD    H,(IX+3)
802B  DD6E02                 LD    L,(IX+2) ; Y-Koordinate
802E  CDC0BB                 CALL  MOVABS ; Grafik-Cursor auf
X,Y-Koordinate
8031  DD5601                 LD    D,(IX+1)
8034  DD5E00                 LD    E,(IX) ; Länge nach de als X-
Offset
8037  D5                     PUSH  DE ; merken
8038  210000                 LD    HL,0 ; Y-Offset
803B  CDF9BB                 CALL  DRAWREL ; waagerechte Linie
ziehen
```

```
803E  E1          POP   HL
803F  E5          PUSH  HL
8040  CDC7BD      CALL  CHGSGN ; Y-Offset negativ
8043  E5          PUSH  HL
8044  110000      LD    DE,0
8047  CDF9BB      CALL  DRAWREL ; senkrechte Linie
ziehen
804A  D1          POP   DE ; negativer X-Offset
804B  210000      LD    HL,0 ; Y-Offset null
804E  CDF9BB      CALL  DRAWREL ; waagerechte Linie
ziehen
8051  E1          POP   HL
8052  110000      LD    DE,0
8055  CDF9BB      CALL  DRAWREL ; senkrechte Linie
ziehen
8058  E1          POP   HL
8059  D1          POP   DE
805A  C3C0BB      JP    MOVABS ; Koordinaten wie-
derherstellen
```

Nachdem das Programm geladen (als Binärfile von Kassette oder Diskette) oder mittels DATA-Lader generiert im Speicher steht, muß es einmalig initialisiert werden werden. Dies geschieht durch Aufruf von CALL &8000. Damit steht der neue Befehl zur Verfügung. Für die Einbindung werden zwei Tabellen benutzt. Die erste, in unserem Beispiel RSX genannt, enthält zuerst die Adresse der zweiten Tabelle, hier TABLE genannt, und anschließend Sprungbefehle auf die eigentliche Erweiterung. Die zweite Tabelle enthält die Namen, unter denen die neuen Befehle aufgerufen werden können. Dabei sind Großbuchstaben sowie Punkte erlaubt. Das letzte Zeichen eines Befehlswort wird durch das gesetzte Bit 7 gekennzeichnet. Danach können weitere Befehlsworte folgen. Das Ende der Tabelle ist durch ein Nullbyte gekennzeichnet. In jeder Tabelle müssen natürlich gleich viel Einträge stehen; zu jedem Befehlswort muß in der ersten die korrespondierende Sprungadresse stehen. Unter dem Label KERNAL müssen wir dem Betriebssystem 4 Bytes zur Verfügung stellen, die zur Verwaltung der Erweiterung benutzt werden. Die 4 Bytes müssen zwischen Adresse &4000 und &BFFF stehen.

Die Routine zum Zeichnen des Quadrats beginnt beim Label QUADRAT. Zuerst wird geprüft, ob drei Parameter übergeben wurden. Ist das nicht der Fall, wird sofort zurückgekehrt. Andernfalls wird die aktuelle Grafikcursorposition geholt und auf den Stack gerettet. Jetzt werden die übergebenen X- und Y-Koordinaten nach DE und HL geholt. Die Basis des Parameterblock steht in IX zur Verfügung. Nachdem der Grafikcursor auf diese Koordinaten gesetzt wird, kann viermal die Routine zum Zeichnen einer Linie relativ zur augenblicklichen Position aufgerufen werden. Um einen negativen Offset zu berechnen, wird die Routine CHGSGN der Inte-

gerarithmetik aufgerufen. Zum Abschluß wird die ursprüngliche Grafikcursorposition wieder hergestellt.

Als Beispiel für eine Anwendung der Routine geben Sie bitte folgendes kleine Programm ein:

```
10 CLS
20 FOR i=35 TO 400 STEP 20
30 IQUADRAT,i,i,30
40 NEXT
```

3.5.3 Der Variablenpointer '@'

Eine besonders für den Maschinenprogrammierer interessante Funktion ist der Variablenpointer, der durch den 'Klammeraffen' aufgerufen wird. Diese Funktion gibt die Adresse zurück, an der eine Variable im Speicher abgelegt ist. Der Aufruf sieht so aus:

```
PRINT @a
```

Damit wird die Adresse der Variablen a ausgegeben. War die Variable noch nicht angelegt, wird die Fehlermeldung 'Improper argument' ausgegeben.

Wenn wir nun an den Inhalt der Variablen wollen, müssen wir zwischen den 3 möglichen Typen unterscheiden.

Bei Integervariablen sieht die Sache am einfachsten aus. An der angegebenen Adresse ist der 16-Bit-Wert abgelegt. Den Wert der Variablen a% erhalten wir also mit folgender Formel:

```
PRINT PEEK(@a%)+256*PEEK(@a%+1)
```

Dabei können wir Werte zwischen 0 und 65535 erhalten. Soll das Vorzeichen noch berücksichtigt werden, müssen wir die Funktion UNT anwenden.

```
PRINT UNT(PEEK(@a%)+256*PEEK(@a%+1))
```

Bei Fließkommavariablen zeigt der Variablenpointer ebenfalls auf den Wert der Variablen, der sich jedoch aus 5 Bytes zusammensetzt. Die ersten 4 Bytes sind die sogenannte Mantisse und das 5. Byte ist der Zweierexponent, mit dem die Mantisse multipliziert werden muß, um den Wert der Variablen zu erhalten. Wenn wir die 4 Mantissenbytes mit m1 bis m4 bezeichnen und den Exponenten mit ex, so ergibt folgende Formel den zugehörigen Fließkommawert:

```
x = (1-2*SGN (m4 AND 128)) * 2↑(ex-129) *
    (1+ ((m4 AND 127)+(m3+(m2+m1/256)/256)/256)/128)
```

Aus der Formel wird deutlich, daß das Vorzeichen der Fließkommazahl im obersten Bit von m4 steckt und daß die Mantissenbytes m1 bis m4 steigende Wertigkeiten haben. Der Zweierexponent enthält einen Offset von 129, so daß sich Werte von 2↑-129 bis 2↑127 ergeben. Probieren wir unsere Formel einmal aus.

```
100 a=-13:' untersuchte Fließkommavariable
110 ad = @a: ' Adresse von a
120 m1=PEEK(ad):m2=PEEK(ad+1):m3=PEEK(ad+2)
130 m4=PEEK(ad+3):ex=PEEK(ad+4)
140 PRINT (1-2*SGN (m4 AND 128)) * 2↑(ex-129) *
    (1+ ((m4 AND 127)+(m3+(m2+m1/256)/256)/256)/128)
```

Wenn Sie das Programm laufen lassen, erhalten Sie den Wert -13 zurück. Ersetzen Sie Zeile 100 einmal durch INPUT a, so können Sie beliebige Werte ausprobieren.

Anwendung findet die Variablenpointer-Funktion im CALL-Befehl, der ja bekanntlich nur 16-Bit-Werte übergeben kann. Wollen Sie also mit Fließkommawerten arbeiten, so können Sie mit '@' die Adresse einer Fließkommazahl übergeben, auf die Sie sich dann beziehen können. Mit dieser Methode ist es natürlich auch möglich, den Wert der Fließkommavariablen direkt zu verändern.

Noch interessanter wird es bei Stringvariablen. Auch hier können wir den Variablenpointer benutzen, der uns wieder die Adresse der Variablen zurückgibt. Dies ist jedoch nicht direkt die Adresse des Strings, sondern des sogenannten Stringdescriptors. Dieser Stringdescriptor ist drei Bytes lang. Das erste Byte enthält die Länge des Strings, also einen Wert von Null bis 255. Die nächsten beiden Bytes enthalten die Adresse des Strings.

```
100 INPUT a$
110 ad=@a$
120 l=PEEK(ad)
130 sa=PEEK(ad+1)+256*PEEK(ad+2)
140 FOR i=sa TO sa+l-1: PRINT CHR$(PEEK(I));:NEXT
```

Das Programm holt Länge und Adresse des Strings, liest und gibt ihn aus.

Auch hier kann über den Variablenpointer ein String an den CALL-Befehl übergeben werden.

Strings lassen sich im Zusammenhang mit dem CALL-Befehl noch ganz anders einsetzen. Man legt dazu ein Maschinenprogramm einfach in einem String ab und kann es mit dem CALL-Befehl und dem Variablenpointer aufrufen. Das Maschinenprogramm muß dazu verschiebbar (keine internen absoluten Sprünge) und darf nicht länger als 255 Bytes sein. Das ist bei kleineren Utilities meist gegegeben. Wollen Sie diese Methode anwenden, so müssen Sie folgendermaßen vorgehen:

Zuerst wird das Maschinenprogramm in die Stringvariable abgelegt. Dies wird meist mit READ und DATA geschehen. Wollen Sie das Programm dann ausführen, so berechnen Sie die Startadresse des Strings (und des Maschinenprogramms) mit dem 'Klammeraffen'.

3.6 Das BASIC-ROM-Listing

3.6.1 Die Fließkommaarithmetik

Sämtliche arithmetische Funktionen, die der BASIC-Interpreter benutzt, stehen im Betriebssystem-ROM. Sie werden über die Sprungtabelle von &BD3D bis &BDC7 aufgerufen. Wenn Sie diese Arithmetikroutinen ändern wollen, so brauchen Sie nur an der entsprechenden Stelle einen Sprung auf Ihre Routine einfügen.

Als ein Beispiel zur Anwendung der Fließkommaroutinen des BASIC-Interpreters zeigen wir Ihnen eine Routine zur Berechnung der Quadratwurzel einer Zahl. Der BASIC-Interpreter des CPC 464 stellt uns diese Funktion zwar schon zur Verfügung, jedoch soll gezeigt werden, daß durch Anwendung von leistungsfähigen Algorithmen diese noch verbessert werden können.

Die eingebaute SQR-Funktion arbeitet nach dem gleichen Algorithmus, nach dem auch die Potenzberechnung geschieht.

SQR(X) = EXP (LOG(X)*0.5)

Es müssen also jedesmal Exponential- und Logarithmusfunktion berechnet werden, was über aufwendige Polynomberechnungen geschieht. Die Quadratwurzel läßt sich jedoch über eine einfache Iterationsvorschrift berechnen.

X(N+1) = (X(N) + A / X(N)) / 2

wobei A die Zahl ist, aus der die Wurzel gezogen werden soll und X(N) der alte und X(N+1) der neue Näherungswert ist. Als Startwert kann man die Zahl A selbst nehmen. Ein besserer Nährungswert ergibt sich jedoch, wenn man den Zweierexponent der Fließkommazahl halbiert. Dann ändert sich das Ergebnis nach vier Iterationen im Rahmen der Rechengenauigkeit nicht mehr. Beachten Sie auch, daß die Division durch 2 nicht als aufwendige Fließkommadivision realisiert wurde, sondern einfach dadurch, daß man den Zweierexponenten um eins erniedrigt. Der Zeitgewinn dieses Verfahrens ist bedeutend. Benötigt die SQR-Routine des Interpreters noch 27 Millisekunden, so schafft unsere Routine die gleiche Aufgabe in knapp 8 Millisekunden; sie ist also mehr als dreimal so schnell.

```
                        ;SCHNELLE SQR-ROUTINE
                        ;LE 18/12/84

AB00                            ORG     &AB00
```

```
BD70            SGN      EQU   &BD70
BD64            DIV      EQU   &BD64
BD58            ADD      EQU   &BD58

AB00 CD70BD     NEWSQR   CALL  SGN ; Vorzeichen prüfen
AB03 3F                  CCF
AB04 C8                  RET   Z ; Null, schon fertig
AB05 F20CAB              JP    P,GOON
AB08 3E01                LD    A,1 ; 'IMPROPER ARGUMENT'
AB0A B7                  OR    A
AB0B C9                  RET

AB0C E5         GOON     PUSH  HL
AB0D 1153AB              LD    DE,STORE1
AB10 010500              LD    BC,5
AB13 EDB0                LDIR  ; Radikant merken
AB15 E1                  POP   HL

AB16 E5                  PUSH  HL
AB17 DDE1                POP   IX
AB19 DD7E04              LD    A,(IX+4) ; Exponent
AB1C D681                SUB   &81 ; normalisieren
AB1E 3F                  CCF
AB1F 1F                  RRA   ; Exponent halbieren
AB20 C601                ADD   A,1
AB22 DD7704              LD    (IX+4),A ; als Startwert

AB25 0604                LD    B,4 ; 4 Iterationen
AB27 C5         ITER     PUSH  BC
AB28 E5                  PUSH  HL
AB29 1158AB              LD    DE,STORE2
AB2C 010500              LD    BC,5
AB2F EDB0                LDIR  ; Näherung merken
AB31 E1                  POP   HL
AB32 E5                  PUSH  HL
AB33 1153AB              LD    DE,STORE1
AB36 EB                  EX    DE,HL
AB37 010500              LD    BC,5
AB3A EDB0                LDIR  ; Radikant holen
AB3C E1                  POP   HL
AB3D 1158AB              LD    DE,STORE2
AB40 CD64BD              CALL  DIV
AB43 1158AB              LD    DE,STORE2
AB46 CD58BD              CALL  ADD
AB49 E5                  PUSH  HL
AB4A DDE1                POP   IX
```

```
AB4C DD3504            DEC    (IX+4) ; ZAHL / 2
AB4F C1                POP    BC
AB50 10D5              DJNZ   ITER
AB52 C9                RET

AB53          STORE1   DEFS   5
AB58          STORE2   DEFS   5
```

Wie kann man aber den Interpreter dazu veranlasssen, die neuen Routine zu verwenden? Für die SQR-Funktion dient der Vektor &BD79. An diese Adresse muß ein Sprung auf unsere Routine eingetragen werden.

JP &AB00

Wenn die Routine von BASIC aus aufgerufen wird, muß das HL-Register auf den Fließkommawert zeigen. Nach Ausführung der Routine muß das HL-Register auf das Ergebnis zeigen. Normalerweise hat sich der Wert dieses Registers nicht verändert. Die Flags zeigen den Fehlerstatus der Funktion an:

Fehlerstatus der Funktionen

C=1	**ordnungsgemäße Ausführung**
C=0 & Z=1	**'Division by zero'**
C=0 & N=1	**'Overflow'**
C=0 & Z=0	**'Improper argument'**

Auf den nächsten Seiten finden Sie das Listing der Fließkommaarithmetik, wobei jede Routine auch die Adresse der Sprungtabelle enthält, über die sie vom BASIC-Interpreter aus angesprochen wird. In Kapitel 3.6.3 finden Sie dann die Integerarithmetik, die vom Interpreter immer dann benutzt wird, wenn dies möglich ist. Da hierbei immer nur mit 2-Byte-Werten gearbeitet wird, ist diese Arithmetik bedeutend schneller als die Rechnung mit Fließkommazahlen. Nutzen Sie dies auch in Ihren Programmen aus und verwenden Sie wenn immer möglich nur Integervariablen. Dies gilt besonders auch für FOR-NEXT-Schleifen (siehe dazu auch Kapitel 3.2).

```
************************************ BD3D  Variable von (de) nach (hl) kopieren
2E18  E5          push  hl
2E19  D5          push  de
2E1A  C5          push  bc
2E1B  EB          ex    de,hl
2E1C  010500      ld    bc,0005       5 Bytes
2E1F  EDB0        ldir                kopieren
2E21  EB          ex    de,hl
2E22  2B          dec   hl
2E23  7E          ld    a,(hl)        a = Exponent
2E24  C1          pop   bc
2E25  D1          pop   de
2E26  E1          pop   hl
2E27  37          scf
2E28  C9          ret

************************************ BD40  Integer nach Fließkomma wandeln
2E29  D5          push  de
2E2A  C5          push  bc
2E2B  F67F        or    7F
2E2D  47          ld    b,a
2E2E  AF          xor   a
2E2F  12          ld    (de),a
2E30  13          inc   de
2E31  12          ld    (de),a
2E32  13          inc   de
2E33  0E90        ld    c,90          Exponent, 2↑15
2E35  7C          ld    a,h
2E36  B7          or    a
2E37  2008        jr    nz,2E41
2E39  4F          ld    c,a
2E3A  65          ld    h,l
2E3B  6F          ld    l,a
2E3C  B4          or    h
2E3D  280D        jr    z,2E4C
2E3F  0E88        ld    c,88          Exponent, 2↑7
2E41  FA4B2E      jp    m,2E4B
2E44  29          add   hl,hl
2E45  0D          dec   c
2E46  B4          or    h
2E47  F2442E      jp    p,2E44
2E4A  7C          ld    a,h
2E4B  A0          and   b
2E4C  EB          ex    de,hl
2E4D  73          ld    (hl),e
2E4E  23          inc   hl
2E4F  77          ld    (hl),a
2E50  23          inc   hl
2E51  71          ld    (hl),c
2E52  C1          pop   bc
2E53  E1          pop   hl
2E54  C9          ret
```

```
********************************** BD43  4-Byte-Wert nach Fließkomma wandeln
2E55  C5          push  bc
2E56  0100A0      ld    bc,A000         Exponent, 2↑31
2E59  CD602E      call  2E60            konvertieren
2E5C  C1          pop   bc
2E5D  C9          ret

******************************* BD94  4-Byte-Wert*256 nach Fließkomma wandeln
2E5E  06A8        ld    b,A8            Exponent, 2↑39
2E60  D5          push  de
2E61  CDA136      call  36A1            Konversion
2E64  D1          pop   de
2E65  C9          ret

****************************************** BD46  Fließkomma nach Integer
2E66  E5          push  hl
2E67  DDE1        pop   ix
2E69  AF          xor   a
2E6A  DD9604      sub   (ix+04)         Exponent
2E6D  281B        jr    z,2E8A          Zahl gleich null ?
2E6F  C690        add   a,90
2E71  D0          ret   nc
2E72  D5          push  de
2E73  C5          push  bc
2E74  C610        add   a,10
2E76  CD3D36      call  363D
2E79  CB21        sla   c
2E7B  ED5A        adc   hl,de
2E7D  2808        jr    z,2E87
2E7F  DD7E03      ld    a,(ix+03)       Vorzeichen der Mantisse
2E82  B7          or    a
2E83  3F          ccf
2E84  C1          pop   bc
2E85  D1          pop   de
2E86  C9          ret

2E87  9F          sbc   a,a
2E88  18F9        jr    2E83

2E8A  6F          ld    l,a
2E8B  67          ld    h,a             Null nach hl
2E8C  37          scf
2E8D  C9          ret

****************************************** BD49  Fließkomma nach Integer
2E8E  CDA12E      call  2EA1            FIX
2E91  D0          ret   nc
2E92  F0          ret   p
2E93  E5          push  hl
2E94  79          ld    a,c
2E95  34          inc   (hl)
2E96  2006        jr    nz,2E9E
2E98  23          inc   hl
2E99  3D          dec   a
2E9A  20F9        jr    nz,2E95
```

```
2E9C  34            inc    (hl)
2E9D  0C            inc    c
2E9E  E1            pop    hl
2E9F  37            scf
2EA0  C9            ret

************************************************** BD4C  FIX
2EA1  E5            push   hl
2EA2  D5            push   de
2EA3  E5            push   hl
2EA4  DDE1          pop    ix
2EA6  CD0436        call   3604         FIX-Funktion
2EA9  D1            pop    de
2EAA  E1            pop    hl
2EAB  C9            ret

************************************************** BD4F  INT
2EAC  CDA12E        call   2EA1         FIX
2EAF  D0            ret    nc
2EB0  C8            ret    z
2EB1  CB78          bit    7,b
2EB3  C8            ret    z
2EB4  18DD          jr     2E93

***************************************************** BD52
2EB6  CDE835        call   35E8         SGN
2EB9  47            ld     b,a
2EBA  2852          jr     z,2F0E
2EBC  FCFB35        call   m,35FB       negativ, dann Vorzeichenwechsel
2EBF  E5            push   hl
2EC0  DD7E04        ld     a,(ix+04)    Exponent
2EC3  D680          sub    80           normalisieren
2EC5  5F            ld     e,a
2EC6  9F            sbc    a,a
2EC7  57            ld     d,a
2EC8  6B            ld     l,e
2EC9  62            ld     h,d
2ECA  29            add    hl,hl
2ECB  29            add    hl,hl
2ECC  29            add    hl,hl
2ECD  19            add    hl,de
2ECE  29            add    hl,hl        mal 77
2ECF  19            add    hl,de
2ED0  29            add    hl,hl
2ED1  29            add    hl,hl
2ED2  19            add    hl,de
2ED3  7C            ld     a,h
2ED4  D609          sub    09
2ED6  5F            ld     e,a
2ED7  E1            pop    hl
2ED8  C5            push   bc
2ED9  D5            push   de
2EDA  C41F2F        call   nz,2F1F      Zahl mit 10↑a multiplizieren
2EDD  FD21132F      ld     iy,2F13      3124999.98
2EE1  CDA035        call   35A0         Vergleich
```

```
2EE4  281B        jr    z,2F01       gleich ?
2EE6  3008        jr    nc,2EF0      größer ?
2EE8  CD1234      call  3412         Multiplikation mit 10
2EEB  D1          pop   de
2EEC  1D          dec   e
2EED  D5          push  de
2EEE  18ED        jr    2EDD

2EF0  FD21182F    ld    iy,2F18      1E9
2EF4  CDA035      call  35A0         Vergleich
2EF7  3808        jr    c,2F01       kleiner ?
2EF9  CD9B34      call  349B         Division durch 10
2EFC  D1          pop   de
2EFD  1C          inc   e
2EFE  D5          push  de
2EFF  18EF        jr    2EF0

2F01  CD8E2E      call  2E8E
2F04  79          ld    a,c
2F05  D1          pop   de
2F06  C1          pop   bc
2F07  4F          ld    c,a
2F08  3D          dec   a
2F09  85          add   a,l
2F0A  6F          ld    l,a
2F0B  D0          ret   nc
2F0C  24          inc   h
2F0D  C9          ret

2F0E  5F          ld    e,a
2F0F  77          ld    (hl),a
2F10  0E01        ld    c,01
2F12  C9          ret

***************************************************************
2F13  F0 1F BC 3E 96                 3124999.98
2F18  FE 27 6B 6E 9E                 1E9

*************************************** BD55  Zahl mit 10↑a multiplizieren
2F1D  2F          cpl   a
2F1E  3C          inc   a
2F1F  B7          or    a
2F20  37          scf
2F21  C8          ret   z
2F22  4F          ld    c,a
2F23  F2282F      jp    p,2F28
2F26  2F          cpl   a
2F27  3C          inc   a
2F28  CD3E2F      call  2F3E
2F2B  2809        jr    z,2F36
2F2D  C5          push  bc
2F2E  F5          push  af
2F2F  CD362F      call  2F36
2F32  F1          pop   af
2F33  C1          pop   bc
```

```
2F34  18F2        jr    2F28

2F36  79          ld    a,c
2F37  B7          or    a
2F38  F29E34      jp    p,349E        Division
2F3B  C31534      jp    3415          Multiplikation

2F3E  118F2F      ld    de,2F8F       1E13
2F41  D60D        sub   0D            -13
2F43  D0          ret   nc            größer gleich ?
2F44  C60C        add   a,0C          + 12
2F46  5F          ld    e,a
2F47  87          add   a,a
2F48  87          add   a,a           mal 5
2F49  83          add   a,e
2F4A  C653        add   a,53
2F4C  5F          ld    e,a           2F53, Zehnerpotenzen
2F4D  CE2F        adc   a,2F
2F4F  93          sub   e
2F50  57          ld    d,a
2F51  AF          xor   a
2F52  C9          ret

******************************************** Fließkommakonstanten
2F53  00 00 00 20 84                  10
2F58  00 00 00 48 87                  100
2F5D  00 00 00 7A 8A                  1000
2F62  00 00 40 1C 8E                  10000
2F67  00 00 50 43 91                  100000
2F6C  00 00 24 74 94                  1000000
2F71  00 80 96 18 98                  10000000
2F76  00 20 BC 3E 9B                  100000000
2F7B  00 28 6B 6E 9E                  1E9
2F80  00 F9 02 15 A2                  1E10
2F85  40 B7 43 3A A5                  1E11
2F8A  10 B5 D4 68 A8                  1E12
2F8F  2A E7 84 11 AC                  1E13

************************************************ BD97   RND Init
2F94  216589      ld    hl,8965
2F97  22E6B8      ld    (B8E6),hl
2F9A  21076C      ld    hl,6C07
2F9D  22E4B8      ld    (B8E4),hl
2FA0  C9          ret

********************************************* BD9A   Random Seed
2FA1  EB          ex    de,hl
2FA2  CD942F      call  2F94          RND Init
2FA5  EB          ex    de,hl
2FA6  CDE835      call  35E8          SGN
2FA9  C8          ret   z
2FAA  11E4B8      ld    de,B8E4       Zeiger auf RND-Mantisse
2FAD  0604        ld    b,04          4 Bytes
2FAF  1A          ld    a,(de)
2FB0  AE          xor   (hl)          neue Mantisse bilden
```

```
2FB1  12              ld     (de),a
2FB2  13              inc    de
2FB3  23              inc    hl
2FB4  10F9            djnz   2FAF           nächstes Byte
2FB6  C9              ret

************************************************** BD9D   RND
2FB7  E5              push   hl
2FB8  2AE6B8          ld     hl,(B8E6)
2FBB  01076C          ld     bc,6C07
2FBE  CDFA2F          call   2FFA
2FC1  E5              push   hl
2FC2  2AE4B8          ld     hl,(B8E4)
2FC5  016589          ld     bc,8965
2FC8  CDFA2F          call   2FFA
2FCB  D5              push   de
2FCC  E5              push   hl
2FCD  2AE6B8          ld     hl,(B8E6)
2FD0  CDFA2F          call   2FFA
2FD3  E3              ex     (sp),hl
2FD4  09              add    hl,bc
2FD5  22E4B8          ld     (B8E4),hl
2FD8  E1              pop    hl
2FD9  01076C          ld     bc,6C07
2FDC  ED4A            adc    hl,bc
2FDE  C1              pop    bc
2FDF  09              add    hl,bc
2FE0  C1              pop    bc
2FE1  09              add    hl,bc
2FE2  22E6B8          ld     (B8E6),hl
2FE5  E1              pop    hl

**************************************** BDA0   letzten RND-Wert holen
2FE6  E5              push   hl
2FE7  DDE1            pop    ix
2FE9  2AE4B8          ld     hl,(B8E4)
2FEC  ED5BE6B8        ld     de,(B8E6)
2FF0  010000          ld     bc,0000
2FF4  DD360480        ld     (ix+04),80     Exponent
2FF7  C3B136          jp     36B1

2FFA  EB              ex     de,hl
2FFB  210000          ld     hl,0000
2FFE  3E11            ld     a,11
3000  3D              dec    a
3001  C8              ret    z
3002  29              add    hl,hl
3003  CB13            rl     e
3005  CB12            rl     d
3007  30F7            jr     nc,3000
3009  09              add    hl,bc
300A  30F4            jr     nc,3000
300C  13              inc    de
300D  18F1            jr     3000
```

```
*********************************************** BD82  LOG10
300F  118B30      ld    de,308B        LOG10(2)
3012  1803        jr    3017

************************************************* BD7F  LOG
3014  118630      ld    de,3086        LOG(2)
3017  CDE835      call  35E8           SGN
301A  3D          dec   a
301B  FE01        cp    01
301D  D0          ret   nc
301E  D5          push  de
301F  CD6C35      call  356C           Exponent testen
3022  F5          push  af
3023  DD360480    ld    (ix+04),80     Exponent, Zahl 0.5 bis 1
3027  118130      ld    de,3081        1/SQR(2)
302A  CD9A35      call  359A           Vergleich
302D  3006        jr    nc,3035        größer ?
302F  DD3404      inc   (ix+04)        Exponent erhöhen, Zahl mal 2
3032  F1          pop   af
3033  3D          dec   a
3034  F5          push  af
3035  CD1633      call  3316           Ergebnis zwischenspeichern
3038  D5          push  de
3039  113233      ld    de,3332        1
303C  CD3F33      call  333F           Addition
303F  EB          ex    de,hl
3040  E1          pop   hl
3041  D5          push  de
3042  113233      ld    de,3332        1
3045  CD3733      call  3337           Subtraktion
3048  D1          pop   de
3049  CD9E34      call  349E           Division
304C  CDA932      call  32A9           Polynomberechnung

************************************** Fließkommakonstanten für LOG
304F  04                               Polynomgrad
3050  4C 4B 57 5E 7F                   0.434259751
3055  0D 08 9B 13 80                   0.576584342
305A  23 93 38 76 80                   0.961800762
305F  20 3B AA 38 82                   2.88539007

************************************************************
3064  D5          push  de
3065  CD1534      call  3415           Multiplikation
3068  D1          pop   de
3069  E3          ex    (sp),hl
306A  7C          ld    a,h
306B  B7          or    a
306C  F27130      jp    p,3071
306F  2F          cpl   a
3070  3C          inc   a
3071  6F          ld    l,a
3072  7C          ld    a,h
3073  2600        ld    h,00
3075  CD292E      call  2E29           Integer nach Fließkomma wandeln
```

```
3078  EB                   ex     de,hl
3079  E1                   pop    hl
307A  CD3F33               call   333F         Addition
307D  D1                   pop    de
307E  C31534               jp     3415         Multiplikation

************************************************************
3081  34 F3 04 35 80                           .707106781 1/SQR(2)
3086  F8 17 72 31 80                           .693147181 LOG(2)
308B  85 9A 20 1A 7F                           .301029996 LOG10(2)

****************************************************** BD85  EXP
3090  06E1                 ld     b,E1
3092  CD0733               call   3307         Exponent vergleichen
3095  D22833               jp     nc,3328      1 als Ergebnis
3098  110031               ld     de,3100      LOG(größte darstellbare Zahl)
309B  CD9A35               call   359A         Vergleich
309E  F2EC36               jp     p,36EC       größer, dann Überlauf
30A1  110531               ld     de,3105      LOG(kleinste darstellbare Zahl)
30A4  CD9A35               call   359A         Vergleich
30A7  FAE636               jp     m,36E6       kleiner, dann Unterlauf, Null
30AA  11FB30               ld     de,30FB      1/LOG(2)
30AD  CDD432               call   32D4
30B0  7B                   ld     a,e
30B1  F2B630               jp     p,30B6
30B4  ED44                 neg    a
30B6  F5                   push   af
30B7  CD1D33               call   331D         multiplizieren
30BA  CD0F33               call   330F         Variable zwischenspeichern
30BD  D5                   push   de
30BE  CDAC32               call   32AC         Polynomberechnung

**************************************** Fließkommakonstanten für EXP
30C1  03                                       Polynomgrad
30C2  F4 32 EB 0F 73                           6.86258E-5
30C7  08 B8 D5 52 7B                           2.57367E-2
30CC  00 00 00 00 80                           0.5

************************************************************
30D1  E3                   ex     (sp),hl
30D2  CDAC32               call   32AC         Polynomberechnung

**************************************** Fließkommakonstanten für EXP
30D5  02                                       Polynomgrad
30D6  09 60 DE 01 78                           1.98164E-3
30DB  F8 17 72 31 7E                           0.173286795
************************************************************
30E0  CD1534               call   3415         Multiplikation
30E3  D1                   pop    de
30E4  E5                   push   hl
30E5  EB                   ex     de,hl
30E6  CD3733               call   3337         Subtraktion
30E9  EB                   ex     de,hl
30EA  E1                   pop    hl
30EB  CD9E34               call   349E         Division
```

```
30EE  11CC30     ld    de,30CC     0.5
30F1  CD3F33     call  333F        Addition
30F4  DD3404     inc   (ix+04)     Exponent erhöhen, Zahl mal 2
30F7  F1         pop   af
30F8  C37B35     jp    357B        Zahl mit 2↑a multiplizieren

*************************************************************
30FB  29 3B AA 38 81               1.44269504 1/LOG(2)
3100  C7 33 0F 30 87               88.0269919 LOG(größte Zahl)
3105  F8 17 72 B1 87               -88.7228391 LOG(kleinste Zahl)
***************************************************** BD79  SQR
310A  11CC30     ld    de,30CC     0.5

************************************************ BD7C  Potenzierung
310D  EB         ex    de,hl
310E  CDE835     call  35E8        SGN, Vorzeichen des Exponents
3111  EB         ex    de,hl
3112  CA2833     jp    z,3328      Null, dann 1 als Ergebnis
3115  F5         push  af
3116  CDE835     call  35E8        SGN, Vorzeichen der Basis
3119  2825       jr    z,3140
311B  47         ld    b,a
311C  FCFB35     call  m,35FB      negativ, dann Vorzeichenwechsel
311F  E5         push  hl
3120  CD8231     call  3182
3123  E1         pop   hl
3124  3825       jr    c,314B
3126  E3         ex    (sp),hl
3127  E1         pop   hl
3128  FA4831     jp    m,3148
312B  C5         push  bc
312C  D5         push  de
312D  CD1430     call  3014        LOG
3130  D1         pop   de
3131  DC1534     call  c,3415      Multiplikation
3134  DC9030     call  c,3090      EXP
3137  C1         pop   bc
3138  D0         ret   nc
3139  78         ld    a,b
313A  B7         or    a
313B  FCFB35     call  m,35FB      negativ, dann Vorzeichenwechsel
313E  37         scf
313F  C9         ret

3140  F1         pop   af
3141  37         scf
3142  F0         ret   p
3143  CDEC36     call  36EC        Überlauf
3146  AF         xor   a
3147  C9         ret

3148  AF         xor   a
3149  3C         inc   a
314A  C9         ret
```

```
314B 4F       ld    c,a
314C F1       pop   af
314D C5       push  bc
314E F5       push  af
314F 79       ld    a,c
3150 37       scf
3151 8F       adc   a,a
3152 30FD     jr    nc,3151
3154 47       ld    b,a
3155 CD0F33   call  330F      Variable zwischenspeichern
3158 EB       ex    de,hl
3159 78       ld    a,b
315A 87       add   a,a
315B 2815     jr    z,3172
315D F5       push  af
315E CD1D33   call  331D      mit Zwischenergebnis multiplizieren
3161 3016     jr    nc,3179
3163 F1       pop   af
3164 30F4     jr    nc,315A
3166 F5       push  af
3167 11E8B8   ld    de,B8E8
316A CD1534   call  3415      Multiplikation
316D 300A     jr    nc,3179
316F F1       pop   af
3170 18E8     jr    315A

3172 F1       pop   af
3173 37       scf
3174 FCFD32   call  m,32FD    Kehrwert bilden
3177 18BE     jr    3137

3179 F1       pop   af
317A F1       pop   af
317B C1       pop   bc
317C FAE636   jp    m,36E6    Unterlauf, Null
317F C3EE36   jp    36EE      Überlauf

3182 C5       push  bc
3183 CD1733   call  3317      Zwischenergebnis holen
3186 CDA12E   call  2EA1      FIX
3189 79       ld    a,c
318A C1       pop   bc
318B 3002     jr    nc,318F
318D 2803     jr    z,3192
318F 78       ld    a,b
3190 B7       or    a
3191 C9       ret

3192 4F       ld    c,a
3193 7E       ld    a,(hl)
3194 1F       rra
3195 9F       sbc   a,a
3196 A0       and   b
3197 47       ld    b,a
3198 79       ld    a,c
```

```
3199  FE02            cp     02
319B  9F              sbc    a,a
319C  D0              ret    nc
319D  7E              ld     a,(hl)
319E  FE27            cp     27
31A0  D8              ret    c
31A1  AF              xor    a
31A2  C9              ret

*************************************************** BD76  PI
31A3  11A931          ld     de,31A9       π
31A6  C3182E          jp     2E18          Variable holen

*************************************************************
31A9  A2 DA 0F 49 82                       3.14159265 π

*********************************************** BD73  DEG/RAD
31AE  32F7B8          ld     (B8F7),a
31B1  C9              ret

************************************************** BD8B  COS
31B2  CDE835          call   35E8          SGN
31B5  FCFB35          call   m,35FB        negativ, dann Vorzeichenwechsel
31B8  F601            or     01
31BA  1801            jr     31BD

*************************************************** BD88  SIN
31BC  AF              xor    a
31BD  F5              push   af
31BE  111D32          ld     de,321D       1/π
31C1  06F0            ld     b,F0
31C3  3AF7B8          ld     a,(B8F7)      DEG ?
31C6  B7              or     a
31C7  2805            jr     z,31CE
31C9  112232          ld     de,3222       1/180
31CC  06F6            ld     b,F6
31CE  CD0733          call   3307          Exponent vergleichen
31D1  303A            jr     nc,320D
31D3  F1              pop    af
31D4  CDD532          call   32D5
31D7  D0              ret    nc
31D8  7B              ld     a,e
31D9  1F              rra
31DA  DCFB35          call   c,35FB        Vorzeichenwechsel
31DD  06E8            ld     b,E8
31DF  CD0733          call   3307          Exponent vergleichen
31E2  D2E636          jp     nc,36E6       Unterlauf, Null
31E5  DD3404          inc    (ix+04)       Exponent erhöhen, Zahl mal 2
31E8  CDA932          call   32A9          Polynomberechnung

************************************* Fließkommakonstanten für SIN
31EB  06                                   Polynomgrad
31EC  1B 2D 1A E6 6E                       -3.42879E-6
31F1  F8 FB 07 28 74                        1.60247E-4
31F6  01 89 68 99 79                       -4.68165E-3
```

```
31FB  E1 DF 35 23 7D                      7.96926E-2
3200  28 E7 5D A5 80                      -0.645964095
3205  A2 DA 0F 49 81                      1.57079633 π/2

*****************************************************
320A  C31534        jp    3415            Multiplikation

320D  F1            pop   af
320E  C22833        jp    nz,3328         SIN ?, dann 1 als Ergebnis
3211  3AF7B8        ld    a,(B8F7)
3214  FE01          cp    01              DEG ?
3216  D8            ret   c               nein, fertig
3217  112732        ld    de,3227         mit π/180
321A  C31534        jp    3415            multiplizieren

*****************************************************
321D  6E 83 F9 22 7F                      0.318309886    1/π
3222  B6 60 0B 36 79                      5.55556E-3     1/180
3227  13 35 FA 0E 7B                      1.74533E-2     π/180
322C  D3 E0 2E 65 86                      57.2957795     180/π

********************************************** BD8E  TAN
3231  CD0F33        call  330F            Zahl zwischenspeichern
3234  D5            push  de
3235  CDB231        call  31B2            COS
3238  E3            ex    (sp),hl
3239  DCBC31        call  c,31BC          SIN
323C  D1            pop   de
323D  DA9E34        jp    c,349E          Division
3240  C9            ret

********************************************** BD91  ATN
3241  CDE835        call  35E8            SGN
3244  F5            push  af
3245  FCFB35        call  m,35FB          negativ, dann Vorzeichenwechsel
3248  06F0          ld    b,F0
324A  CD0733        call  3307            Exponent vergleichen
324D  304A          jr    nc,3299
324F  3D            dec   a
3250  F5            push  af
3251  F4FD32        call  p,32FD          Kehrwert bilden
3254  CDA932        call  32A9            Polynomberechnung

***************************************** Fließkommakonstanten für ATN
3257  0B                                  Polynomgrad
3258  FF C1 03 0F 77                      1.09112E-3
325D  83 FC E8 EB 79                      -7.19941E-2
3262  6F CA 78 36 7B                      2.22744E-2
3267  D5 3E B0 B5 7C                      -4.43575E-2
326C  B0 C1 8B 09 7D                      6.71611E-2
3271  AF E8 32 B4 7D                      -8.79877E-2
3276  74 6C 65 62 7D                      0.110545013
327B  D1 F5 37 92 7E                      -0.142791596
3280  7A C3 CB 4C 7E                      0.199996046
3285  83 A7 AA AA 7F                      -0.333333239
```

```
328A  FE FF FF FF 7F                          0.5

************************************************************
328F  CD1534      call   3415        Multiplikation
3292  F1          pop    af
3293  110532      ld     de,3205     π/2
3296  F43B33      call   p,333B      Subtraktion
3299  3AF7B8      ld     a,(B8F7)    DEG ?
329C  B7          or     a
329D  112C32      ld     de,322C     180/π
32A0  C41534      call   nz,3415     falls DEG dann multiplizieren
32A3  F1          pop    af
32A4  FCFB35      call   m,35FB      negativ, dann Vorzeichenwechsel
32A7  37          scf
32A8  C9          ret

****************************************************** Polynomberechnung
32A9  CD1D33      call   331D        multiplizieren
32AC  CD1633      call   3316        Variable zwischenspeichern
32AF  EB          ex     de,hl
32B0  D1          pop    de
32B1  1A          ld     a,(de)      Polynomgrad holen
32B2  13          inc    de
32B3  47          ld     b,a         nach b
32B4  CD182E      call   2E18        Variable holen
32B7  13          inc    de
32B8  13          inc    de
32B9  13          inc    de          plus 5, nächster Koeffizient
32BA  13          inc    de
32BB  13          inc    de
32BC  D5          push   de
32BD  11EDB8      ld     de,B8ED     Zwischenspeicher
32C0  05          dec    b           nächster Koeffizient
32C1  C8          ret    z
32C2  C5          push   bc
32C3  11F2B8      ld     de,B8F2     Zwischenspeicher
32C6  CD1534      call   3415        Multiplikation
32C9  C1          pop    bc
32CA  D1          pop    de
32CB  D5          push   de
32CC  C5          push   bc
32CD  CD3F33      call   333F        Addition
32D0  C1          pop    bc
32D1  D1          pop    de
32D2  18E3        jr     32B7

************************************************************
32D4  AF          xor    a
32D5  F5          push   af
32D6  CD1534      call   3415        Multiplikation
32D9  F1          pop    af
32DA  11CC30      ld     de,30CC     0.5
32DD  C43F33      call   nz,333F     Addition
32E0  E5          push   hl
32E1  CD662E      call   2E66        Fließkomma nach Integer
```

```
32E4  3013        jr    nc,32F9
32E6  D1          pop   de
32E7  E5          push  hl
32E8  F5          push  af
32E9  D5          push  de
32EA  11EDB8      ld    de,B8ED
32ED  CD292E      call  2E29        Integer nach Fließkomma wandeln
32F0  EB          ex    de,hl
32F1  E1          pop   hl
32F2  CD3733      call  3337        Subtraktion
32F5  F1          pop   af
32F6  D1          pop   de
32F7  37          scf
32F8  C9          ret

32F9  E1          pop   hl
32FA  AF          xor   a
32FB  3C          inc   a
32FC  C9          ret

*************************************************** Kehrwert bilden
32FD  CD1633      call  3316        Variable zwischenspeichern
3300  EB          ex    de,hl
3301  CD2833      call  3328        1 holen
3304  C39E34      jp    349E        Division

*********************************************** Exponent vergleichen
3307  CD6C35      call  356C
330A  F0          ret   p
330B  B8          cp    b
330C  C8          ret   z
330D  3F          ccf
330E  C9          ret

****************************************** Variable zwischenspeichern
330F  EB          ex    de,hl
3310  21E8B8      ld    hl,B8E8     Zieladresse
3313  C3182E      jp    2E18        Variable kopieren

****************************************** Variable zwischenspeichern
3316  EB          ex    de,hl
3317  21F2B8      ld    hl,B8F2
331A  C3182E      jp    2E18        Variable holen

*********************************************************************
331D  EB          ex    de,hl
331E  21EDB8      ld    hl,B8ED
3321  CD182E      call  2E18        Variable holen
3324  EB          ex    de,hl
3325  C31534      jp    3415        Multiplikation

************************************************ Konstante 1 holen
3328  D5          push  de
3329  113233      ld    de,3332     1
332C  CD182E      call  2E18        Variable holen
```

```
332F  D1              pop   de
3330  37              scf
3331  C9              ret

**************************************************************
3332  00 00 00 00 81                      1

************************************* BD5B  Subtraktion  (hl) := (hl) - (de)
3337  3E01            ld    a,01
3339  1805            jr    3340

************************************* BD5E  Subtraktion  (hl) := (de) - (hl)
333B  3E80            ld    a,80
333D  1801            jr    3340

*************************************** BD58  Addition  (hl) := (hl) + (de)
333F  AF              xor   a               Carry löschen
3340  E5              push  hl
3341  DDE1            pop   ix
3343  D5              push  de
3344  FDE1            pop   iy
3346  DD4603          ld    b,(ix+03)       Vorzeichen erster Operand
3349  FD4E03          ld    c,(iy+03)       Vorzeichen zweiter Operand
334C  B7              or    a
334D  280B            jr    z,335A
334F  FA5833          jp    m,3358
3352  3E80            ld    a,80
3354  A9              xor   c
3355  4F              ld    c,a
3356  1802            jr    335A

3358  A8              xor   b
3359  47              ld    b,a
335A  DD7E04          ld    a,(ix+04)       Exponenten
335D  FDBE04          cp    (iy+04)         vergleichen
3360  3014            jr    nc,3376
3362  50              ld    d,b
3363  41              ld    b,c
3364  4A              ld    c,d
3365  B7              or    a
3366  57              ld    d,a
3367  FD7E04          ld    a,(iy+04)       Exponent
336A  DD7704          ld    (ix+04),a       Exponent
336D  2854            jr    z,33C3
336F  92              sub   d
3370  FE21            cp    21
3372  304F            jr    nc,33C3
3374  1811            jr    3387

3376  AF              xor   a
3377  FD9604          sub   (iy+04)         Exponent
337A  2859            jr    z,33D5
337C  DD8604          add   a,(ix+04)       Exponent
337F  FE21            cp    21
3381  3052            jr    nc,33D5
```

```
3383  E5          push  hl
3384  FDE1        pop   iy
3386  EB          ex    de,hl
3387  5F          ld    e,a
3388  78          ld    a,b
3389  A9          xor   c
338A  F5          push  af
338B  C5          push  bc
338C  7B          ld    a,e
338D  CD4336      call  3643
3390  79          ld    a,c
3391  C1          pop   bc
3392  4F          ld    c,a
3393  F1          pop   af
3394  FADA33      jp    m,33DA
3397  FD7E00      ld    a,(iy+00)
339A  85          add   a,l
339B  6F          ld    l,a
339C  FD7E01      ld    a,(iy+01)
339F  8C          adc   a,h
33A0  67          ld    h,a
33A1  FD7E02      ld    a,(iy+02)
33A4  8B          adc   a,e
33A5  5F          ld    e,a
33A6  FD7E03      ld    a,(iy+03)
33A9  CBFF        set   7,a
33AB  8A          adc   a,d
33AC  57          ld    d,a
33AD  D2BA36      jp    nc,36BA
33B0  CB1A        rr    d
33B2  CB1B        rr    e
33B4  CB1C        rr    h
33B6  CB1D        rr    l
33B8  CB19        rr    c
33BA  DD3404      inc   (ix+04)       Exponent erhöhen
33BD  C2BA36      jp    nz,36BA
33C0  C3EE36      jp    36EE          Überlauf

*************************************************************
33C3  FD7E02      ld    a,(iy+02)
33C6  DD7702      ld    (ix+02),a
33C9  FD7E01      ld    a,(iy+01)
33CC  DD7701      ld    (ix+01),a
33CF  FD7E00      ld    a,(iy+00)
33D2  DD7700      ld    (ix+00),a
33D5  DD7003      ld    (ix+03),b
33D8  37          scf
33D9  C9          ret

33DA  AF          xor   a
33DB  91          sub   c
33DC  4F          ld    c,a
33DD  FD7E00      ld    a,(iy+00)
33E0  9D          sbc   a,l
33E1  6F          ld    l,a
```

```
33E2  FD7E01      ld    a,(iy+01)
33E5  9C          sbc   a,h
33E6  67          ld    h,a
33E7  FD7E02      ld    a,(iy+02)
33EA  9B          sbc   a,e
33EB  5F          ld    e,a
33EC  FD7E03      ld    a,(iy+03)
33EF  CBFF        set   7,a
33F1  9A          sbc   a,d
33F2  57          ld    d,a
33F3  3016        jr    nc,340B
33F5  78          ld    a,b
33F6  2F          cpl   a
33F7  47          ld    b,a
33F8  AF          xor   a
33F9  91          sub   c
33FA  4F          ld    c,a
33FB  3E00        ld    a,00
33FD  9D          sbc   a,l
33FE  6F          ld    l,a
33FF  3E00        ld    a,00
3401  9C          sbc   a,h
3402  67          ld    h,a
3403  3E00        ld    a,00
3405  9B          sbc   a,e
3406  5F          ld    e,a
3407  3E00        ld    a,00
3409  9A          sbc   a,d
340A  57          ld    d,a
340B  87          add   a,a
340C  DABA36      jp    c,36BA
340F  C3B136      jp    36B1

*********************************************** Multiplikation mit
3412  11532F      ld    de,2F53          10

********************************************** BD61  Multiplikation
3415  D5          push  de
3416  FDE1        pop   iy
3418  E5          push  hl
3419  DDE1        pop   ix
341B  FD7E04      ld    a,(iy+04)        Exponent
341E  B7          or    a
341F  282C        jr    z,344D
3421  3D          dec   a
3422  CD4835      call  3548
3425  2826        jr    z,344D
3427  3021        jr    nc,344A
3429  F5          push  af
342A  C5          push  bc
342B  CD5034      call  3450
342E  79          ld    a,c
342F  C1          pop   bc
3430  4F          ld    c,a
3431  F1          pop   af
```

```
3432 CB7A      bit   7,d
3434 200D      jr    nz,3443
3436 3D        dec   a
3437 2814      jr    z,344D
3439 CB21      sla   c
343B CB15      rl    l
343D CB14      rl    h
343F CB13      rl    e
3441 CB12      rl    d
3443 DD7704    ld    (ix+04),a      Exponent
3446 B7        or    a
3447 C2BA36    jp    nz,36BA
344A C3EE36    jp    36EE           Überlauf

344D C3E636    jp    36E6           Unterlauf

*********************************************************************
3450 210000    ld    hl,0000
3453 5D        ld    e,l
3454 54        ld    d,h
3455 FD7E00    ld    a,(iy+00)
3458 CD9334    call  3493
345B FD7E01    ld    a,(iy+01)
345E CD9334    call  3493
3461 FD7E02    ld    a,(iy+02)
3464 CD9334    call  3493
3467 FD7E03    ld    a,(iy+03)
346A F680      or    80
346C 0608      ld    b,08
346E 1F        rra
346F 4F        ld    c,a
3470 3014      jr    nc,3486
3472 7D        ld    a,l
3473 DD8600    add   a,(ix+00)
3476 6F        ld    l,a
3477 7C        ld    a,h
3478 DD8E01    adc   a,(ix+01)
347B 67        ld    h,a
347C 7B        ld    a,e
347D DD8E02    adc   a,(ix+02)
3480 5F        ld    e,a
3481 7A        ld    a,d
3482 DD8E03    adc   a,(ix+03)
3485 57        ld    d,a
3486 CB1A      rr    d
3488 CB1B      rr    e
348A CB1C      rr    h
348C CB1D      rr    l
348E CB19      rr    c
3490 10DE      djnz  3470
3492 C9        ret
```

```
3493  B7              or     a
3494  20D6            jr     nz,346C
3496  6C              ld     l,h
3497  63              ld     h,e
3498  5A              ld     e,d
3499  57              ld     d,a
349A  C9              ret

****************************************************  Division durch 10
349B  11532F          ld     de,2F53

****************************************************  BD64  Division
349E  D5              push   de
349F  FDE1            pop    iy
34A1  E5              push   hl
34A2  DDE1            pop    ix
34A4  AF              xor    a
34A5  FD9604          sub    (iy+04)           Exponent
34A8  2858            jr     z,3502
34AA  CD4835          call   3548
34AD  CAE636          jp     z,36E6            Unterlauf
34B0  304D            jr     nc,34FF
34B2  C5              push   bc
34B3  4F              ld     c,a
34B4  5E              ld     e,(hl)
34B5  23              inc    hl
34B6  56              ld     d,(hl)
34B7  23              inc    hl
34B8  7E              ld     a,(hl)
34B9  23              inc    hl
34BA  66              ld     h,(hl)
34BB  6F              ld     l,a
34BC  EB              ex     de,hl
34BD  FD4603          ld     b,(iy+03)
34C0  CBF8            set    7.b
34C2  CD3235          call   3532
34C5  3006            jr     nc,34CD
34C7  79              ld     a,c
34C8  B7              or     a
34C9  2008            jr     nz,34D3
34CB  1831            jr     34FE

34CD  0D              dec    c
34CE  29              add    hl,hl
34CF  CB13            rl     e
34D1  CB12            rl     d
34D3  DD7104          ld     (ix+04),c         Exponent
34D6  CD0735          call   3507
34D9  DD7103          ld     (ix+03),c
34DC  CD0735          call   3507
34DF  DD7102          ld     (ix+02),c
34E2  CD0735          call   3507
34E5  DD7101          ld     (ix+01),c
34E8  CD0735          call   3507
34EB  D43235          call   nc,3532
```

```
34EE  9F          sbc   a,a
34EF  69          ld    l,c
34F0  DD6601      ld    h,(ix+01)
34F3  DD5E02      ld    e,(ix+02)
34F6  DD5603      ld    d,(ix+03)
34F9  C1          pop   bc
34FA  4F          ld    c,a
34FB  C3BA36      jp    36BA

34FE  C1          pop   bc
34FF  C3EE36      jp    36EE          Überlauf

3502  CD9435      call  3594
3505  AF          xor   a
3506  C9          ret

3507  0E01        ld    c,01
3509  3808        jr    c,3513
350B  7A          ld    a,d
350C  B8          cp    b
350D  3F          ccf
350E  CC3635      call  z,3536
3511  3013        jr    nc,3526
3513  7D          ld    a,l
3514  FD9600      sub   (iy+00)
3517  6F          ld    l,a
3518  7C          ld    a,h
3519  FD9E01      sbc   a,(iy+01)
351C  67          ld    h,a
351D  7B          ld    a,e
351E  FD9E02      sbc   a,(iy+02)
3521  5F          ld    e,a
3522  7A          ld    a,d
3523  98          sbc   a,b
3524  57          ld    d,a
3525  37          scf
3526  CB11        rl    c
3528  9F          sbc   a,a
3529  29          add   hl,hl
352A  CB13        rl    e
352C  CB12        rl    d
352E  3C          inc   a
352F  20D8        jr    nz,3509
3531  C9          ret

3532  7A          ld    a,d
3533  B8          cp    b
3534  3F          ccf
3535  C0          ret   nz
3536  7B          ld    a,e
3537  FDBE02      cp    (iy+02)
353A  3F          ccf
353B  C0          ret   nz
353C  7C          ld    a,h
353D  FDBE01      cp    (iy+01)
```

```
3540  3F          ccf
3541  C0          ret    nz
3542  7D          ld     a,l
3543  FDBE00      cp     (iy+00)
3546  3F          ccf
3547  C9          ret

3548  4F          ld     c,a
3549  DD7E03      ld     a,(ix+03)
354C  FDAE03      xor    (iy+03)
354F  47          ld     b,a
3550  DD7E04      ld     a,(ix+04)         Exponent
3553  B7          or     a
3554  C8          ret    z
3555  81          add    a,c
3556  4F          ld     c,a
3557  1F          rra
3558  A9          xor    c
3559  79          ld     a,c
355A  F26835      jp     p,3568
355D  DDCB03FE    set    7,(ix+03)         Vorzeichen negativ
3561  D67F        sub    7F
3563  37          scf
3564  C0          ret    nz
3565  FE01        cp     01
3567  C9          ret

3568  B7          or     a
3569  F8          ret    m
356A  AF          xor    a
356B  C9          ret

356C  E5          push   hl
356D  DDE1        pop    ix
356F  DD7E04      ld     a,(ix+04)         Exponent
3572  B7          or     a
3573  C8          ret    z
3574  D680        sub    80
3576  37          scf
3577  C9          ret

****************************************** BD67   mit 2↑a multiplizieren
3578  E5          push   hl
3579  DDE1        pop    ix
357B  B7          or     a                 Zweierexponent im Akku
357C  FA8935      jp     m,3589            negativ ?
357F  DD8604      add    a,(ix+04)         Exponent erhöhen
3582  DD7704      ld     (ix+04),a         und wieder abspeichern
3585  3F          ccf
3586  D8          ret    c
3587  180B        jr     3594              Überlauf ?
```

```
3589  DD8604     add   a,(ix+04)     Exponent addieren
358C  3802       jr    c,3590        kein Unterlauf
358E  AF         xor   a             Null als Ergebnis
358F  37         scf
3590  DD7704     ld    (ix+04),a     Exponent wieder abspeichern
3593  C9         ret

3594  DD4603     ld    b,(ix+03)     Vorzeichen der Mantisse
3597  CDEE36     call  36EE          Überlauf

********************************************** BD6A  Vergleich
359A  E5         push  hl
359B  DDE1       pop   ix
359D  D5         push  de
359E  FDE1       pop   iy
35A0  DD7E04     ld    a,(ix+04)     Exponenten
35A3  FDBE04     cp    (iy+04)       vergleichen
35A6  383A       jr    c,35E2
35A8  2033       jr    nz,35DD
35AA  B7         or    a
35AB  C8         ret   z
35AC  DD7E03     ld    a,(ix+03)
35AF  FDAE03     xor   (iy+03)
35B2  FADD35     jp    m,35DD
35B5  DD7E03     ld    a,(ix+03)
35B8  FD9603     sub   (iy+03)
35BB  2017       jr    nz,35D4
35BD  DD7E02     ld    a,(ix+02)
35C0  FD9602     sub   (iy+02)
35C3  200F       jr    nz,35D4
35C5  DD7E01     ld    a,(ix+01)
35C8  FD9601     sub   (iy+01)
35CB  2007       jr    nz,35D4
35CD  DD7E00     ld    a,(ix+00)
35D0  FD9600     sub   (iy+00)
35D3  C8         ret   z
35D4  9F         sbc   a,a
35D5  FDAE03     xor   (iy+03)
35D8  87         add   a,a
35D9  9F         sbc   a,a
35DA  D8         ret   c
35DB  3C         inc   a
35DC  C9         ret

35DD  DD7E03     ld    a,(ix+03)
35E0  18F6       jr    35D8

35E2  FD7E03     ld    a,(iy+03)
35E5  2F         cpl   a
35E6  18F0       jr    35D8

************************************************ BD70  SGN
35E8  E5         push  hl
35E9  DDE1       pop   ix
35EB  DD7E04     ld    a,(ix+04)     Exponent
```

```
35EE  B7            or    a
35EF  C8            ret   z
35F0  DD7E03        ld    a,(ix+03)
35F3  87            add   a,a
35F4  9F            sbc   a,a
35F5  D8            ret   c
35F6  3C            inc   a
35F7  C9            ret

*************************************** BD6D  Vorzeichenwechsel
35F8  E5            push  hl
35F9  DDE1          pop   ix
35FB  DD7E03        ld    a,(ix+03)      Vorzeichen der Mantisse
35FE  EE80          xor   80             invertieren
3600  DD7703        ld    (ix+03),a
3603  C9            ret

***************************************************** FIX
3604  AF            xor   a
3605  DD9604        sub   (ix+04)        Exponent
3608  200A          jr    nz,3614        Zahl ungleich Null, dann nach Integer
360A  0604          ld    b,04
360C  77            ld    (hl),a         Mantisse löschen
360D  23            inc   hl
360E  10FC          djnz  360C
3610  0E01          ld    c,01
3612  37            scf
3613  C9            ret

*********************************** Konvertierung Fließkomma nach Integer
3614  C6A0          add   a,A0
3616  D0            ret   nc
3617  E5            push  hl
3618  CD3D36        call  363D
361B  AF            xor   a
361C  B8            cp    b
361D  8F            adc   a,a
361E  B1            or    c
361F  4D            ld    c,l
3620  44            ld    b,h
3621  E1            pop   hl
3622  71            ld    (hl),c
3623  23            inc   hl
3624  70            ld    (hl),b
3625  23            inc   hl
3626  73            ld    (hl),e
3627  23            inc   hl
3628  5F            ld    e,a
3629  7E            ld    a,(hl)
362A  72            ld    (hl),d
362B  E680          and   80
362D  47            ld    b,a
362E  0E04          ld    c,04
3630  AF            xor   a
3631  B6            or    (hl)
```

```
3632  2005      jr   nz,3639
3634  2B        dec  hl
3635  0D        dec  c
3636  20F9      jr   nz,3631
3638  0C        inc  c
3639  7B        ld   a,e
363A  B7        or   a
363B  37        scf
363C  C9        ret

363D  FE21      cp   21
363F  3802      jr   c,3643
3641  3E21      ld   a,21
3643  5E        ld   e,(hl)
3644  23        inc  hl
3645  56        ld   d,(hl)
3646  23        inc  hl
3647  4E        ld   c,(hl)
3648  23        inc  hl
3649  66        ld   h,(hl)
364A  69        ld   l,c
364B  EB        ex   de,hl
364C  CBFA      set  7,d
364E  010000    ld   bc,0000
3651  180B      jr   365E

3653  4F        ld   c,a
3654  78        ld   a,b
3655  B5        or   l
3656  47        ld   b,a
3657  79        ld   a,c
3658  4D        ld   c,l
3659  6C        ld   l,h
365A  63        ld   h,e
365B  5A        ld   e,d
365C  1600      ld   d,00
365E  D608      sub  08
3660  30F1      jr   nc,3653
3662  C608      add  a,08
3664  C8        ret  z
3665  CB3A      srl  d
3667  CB1B      rr   e
3669  CB1C      rr   h
366B  CB1D      rr   l
366D  CB19      rr   c
366F  3D        dec  a
3670  20F3      jr   nz,3665
3672  C9        ret

3673  14        inc  d
3674  15        dec  d
3675  F8        ret  m
3676  2017      jr   nz,368F
3678  57        ld   d,a
3679  7B        ld   a,e
```

```
367A  B4        or    h
367B  B5        or    l
367C  B1        or    c
367D  C8        ret   z
367E  7A        ld    a,d
367F  D608      sub   08
3681  381C      jr    c,369F
3683  C8        ret   z
3684  53        ld    d,e
3685  5C        ld    e,h
3686  65        ld    h,l
3687  69        ld    l,c
3688  0E00      ld    c,00
368A  14        inc   d
368B  15        dec   d
368C  28F1      jr    z,367F
368E  F8        ret   m
368F  3D        dec   a
3690  C8        ret   z
3691  CB21      sla   c
3693  CB15      rl    l
3695  CB14      rl    h
3697  CB13      rl    e
3699  CB12      rl    d
369B  F28F36    jp    p,368F
369E  C9        ret

369F  AF        xor   a
36A0  C9        ret

************************************ Konvertierung Integer nach Fließkomma
36A1  E5        push  hl
36A2  DDE1      pop   ix
36A4  DD7004    ld    (ix+04),b     Exponent
36A7  47        ld    b,a
36A8  5E        ld    e,(hl)
36A9  23        inc   hl
36AA  56        ld    d,(hl)
36AB  23        inc   hl
36AC  7E        ld    a,(hl)
36AD  23        inc   hl
36AE  66        ld    h,(hl)
36AF  6F        ld    l,a
36B0  EB        ex    de,hl
36B1  DD7E04    ld    a,(ix+04)     Exponent
36B4  CD7336    call  3673
36B7  DD7704    ld    (ix+04),a     Exponent
36BA  CB21      sla   c
36BC  3013      jr    nc,36D1
36BE  2C        inc   l
36BF  2010      jr    nz,36D1
36C1  24        inc   h
36C2  200D      jr    nz,36D1
36C4  1C        inc   e
36C5  200A      jr    nz,36D1
```

```
36C7 14         inc   d
36C8 2007       jr    nz,36D1
36CA DD3404     inc   (ix+04)      Exponent
36CD 281F       jr    z,36EE       Überlauf
36CF 1680       ld    d,80
36D1 78         ld    a,b
36D2 F67F       or    7F
36D4 A2         and   d
36D5 DD7703     ld    (ix+03),a
36D8 DD7302     ld    (ix+02),e
36DB DD7401     ld    (ix+01),h
36DE DD7500     ld    (ix+00),l
36E1 DDE5       push  ix
36E3 E1         pop   hl
36E4 37         scf
36E5 C9         ret

**************************************************** Unterlauf, Null
36E6 AF         xor   a
36E7 DD7704     ld    (ix+04),a    Exponent
36EA 18F5       jr    36E1

****************************************** Überlauf, größte positive Zahl
36EC 0600       ld    b,00         Vorzeichen positiv
36EE 78         ld    a,b
36EF F67F       or    7F
36F1 DD7703     ld    (ix+03),a    Mantisse mit Vorzeichen
36F4 F6FF       or    FF
36F6 DD7704     ld    (ix+04),a    Exponent
36F9 DD7700     ld    (ix+00),a
36FC DD7701     ld    (ix+01),a
36FF DD7702     ld    (ix+02),a
3702 C9         ret
3703 C7         rst   0
3704 C7         rst   0
3705 C7         rst   0
3706 C7         rst   0
3707 C7         rst   0
```

```
*********************************************************** BDA3
3708  44          ld    b,h      Vorzeichen merken
3709  CDD137      call  37D1     Absolutwert bilden
370C  1802        jr    3710

*********************************************************** BDA6
370E  0600        ld    b,00
3710  1E00        ld    e,00
3712  0E02        ld    c,02
3714  C9          ret

**************************************** BDA9  Vorzeichen in b übernehmen
3715  7C          ld    a,h
3716  B7          or    a
3717  FA2037      jp    m,3720
371A  B0          or    b        Vorzeichen des Ergebnisses
371B  FAD437      jp    m,37D4   Vorzeichenwechsel
371E  37          scf
371F  C9          ret

3720  EE80        xor   80       Vorzeichenbit umkehren
3722  B5          or    l
3723  C0          ret   nz
3724  78          ld    a,b
3725  37          scf
3726  8F          adc   a,a
3727  C9          ret

******************************************* BDAC  Addition hl := hl + de
3728  B7          or    a        Carry-Flag löschen
3729  ED5A        adc   hl,de    Addition
372B  37          scf
372C  E0          ret   po       Ergebnis positiv ?
372D  F6FF        or    FF       Flags setzen
372F  C9          ret

****************************************** BDB2  Subtraktion hl := de - hl
3730  EB          ex    de,hl    Operanden vertauschen

****************************************** BDAF  Subtraktion hl := hl - de
3731  B7          or    a        Carry-Flag löschen
3732  ED52        sbc   hl,de    Subtraktion
3734  37          scf
3735  E0          ret   po       Ergebnis positiv ?
3736  F6FF        or    FF       Flags setzen
3738  C9          ret

**************************************** BDB5  Multiplikation mit Vorzeichen
3739  CD4537      call  3745     Vorzeichen des Ergebnisses bestimmen
373C  CD5037      call  3750     vorzeichenlose Multiplikation
373F  D21537      jp    nc,3715  Vorzeichen übernehmen
3742  F6FF        or    FF
3744  C9          ret
```

```
************************************ Vorzeichen des Ergebnisses bestimmen
3745  7C          ld    a,h        Vorzeichen von hl
3746  AA          xor   d          und Vorzeichen von de
3747  47          ld    b,a        nach b bringen
3748  EB          ex    de,hl
3749  CDD137      call  37D1       Absolutwert von de bilden
374C  EB          ex    de,hl
374D  C3D137      jp    37D1       Absolutwert von hl bilden

************************************ BDBE  Multiplikation ohne Vorzeichen
3750  7C          ld    a,h
3751  B7          or    a
3752  2805        jr    z,3759
3754  7A          ld    a,d
3755  B7          or    a
3756  37          scf
3757  C0          ret   nz
3758  EB          ex    de,hl
3759  B5          or    l
375A  C8          ret   z
375B  7A          ld    a,d
375C  B3          or    e
375D  7D          ld    a,l
375E  6B          ld    l,e
375F  62          ld    h,d
3760  C8          ret   z
3761  FE03        cp    03
3763  3810        jr    c,3775
3765  37          scf
3766  8F          adc   a,a
3767  30FD        jr    nc,3766
3769  29          add   hl,hl
376A  D8          ret   c
376B  87          add   a,a
376C  3002        jr    nc,3770
376E  19          add   hl,de
376F  D8          ret   c
3770  FE80        cp    80
3772  20F5        jr    nz,3769
3774  C9          ret

3775  FE01        cp    01
3777  C8          ret   z
3778  29          add   hl,hl
3779  C9          ret

****************************************** BDB8  Division mit Vorzeichen
377A  CD8937      call  3789       Division hl := hl/de
377D  DA1537      jp    c,3715     Vorzeichen übernehmen
3780  C9          ret

*********************************************** BDBB  MOD
3781  4C          ld    c,h        Vorzeichen merken
3782  CD8937      call  3789       Division
3785  EB          ex    de,hl      Rest nach hl
```

```
3786  41        ld    b,c          Vorzeichen zurückholen
3787  18F4      jr    377D         und übernehmen

3789  CD4537    call  3745         Vorzeichen des Ergebnisses bestimmen

************************************* BDC1  Division hl := hl/de, de := Rest
378C  7A        ld    a,d          ohne Vorzeichen
378D  B3        or    e            Divisor Null, dann beenden
378E  C8        ret   z
378F  C5        push  bc
3790  EB        ex    de,hl
3791  0601      ld    b,01
3793  7C        ld    a,h
3794  B7        or    a
3795  2009      jr    nz,37A0
3797  7A        ld    a,d
3798  BD        cp    l
3799  3805      jr    c,37A0
379B  65        ld    h,l
379C  2E00      ld    l,00
379E  0609      ld    b,09
37A0  7B        ld    a,e
37A1  95        sub   l
37A2  7A        ld    a,d
37A3  9C        sbc   a,h
37A4  3805      jr    c,37AB
37A6  04        inc   b
37A7  29        add   hl,hl
37A8  30F6      jr    nc,37A0
37AA  3F        ccf
37AB  3F        ccf
37AC  78        ld    a,b
37AD  44        ld    b,h
37AE  4D        ld    c,l
37AF  210000    ld    hl,0000
37B2  3D        dec   a
37B3  2003      jr    nz,37B8
37B5  1817      jr    37CE

37B7  29        add   hl,hl
37B8  F5        push  af
37B9  78        ld    a,b
37BA  1F        rra
37BB  47        ld    b,a
37BC  79        ld    a,c
37BD  1F        rra
37BE  4F        ld    c,a
37BF  7B        ld    a,e
37C0  91        sub   c
37C1  7A        ld    a,d
37C2  98        sbc   a,b
37C3  3805      jr    c,37CA
37C5  57        ld    d,a
37C6  7B        ld    a,e
37C7  91        sub   c
```

```
37C8  5F              ld    e,a
37C9  2C              inc   l
37CA  F1              pop   af
37CB  3D              dec   a
37CC  20E9            jr    nz,37B7
37CE  37              scf
37CF  C1              pop   bc
37D0  C9              ret

*******************************************  Absolutwert bilden
37D1  7C              ld    a,h          Vorzeichen testen
37D2  B7              or    a
37D3  F0              ret   p            positiv, dann schon fertig

**************************************** BDC7  Vorzeichenwechsel hl
37D4  AF              xor   a
37D5  95              sub   l
37D6  6F              ld    l,a
37D7  9C              sbc   a,h
37D8  95              sub   l
37D9  BC              cp    h
37DA  67              ld    h,a
37DB  37              scf
37DC  C0              ret   nz
37DD  FE01            cp    01
37DF  C9              ret

*************************************** BDCA  SGN  Vorzeichen von hl
37E0  7C              ld    a,h
37E1  87              add   a,a
37E2  9F              sbc   a,a
37E3  D8              ret   c
37E4  B5              or    l
37E5  C8              ret   z
37E6  AF              xor   a
37E7  3C              inc   a
37E8  C9              ret

***************************************** BDC4  Vergleich hl <> de
37E9  7C              ld    a,h          Vorzeichen von hl
37EA  AA              xor   d            und Vorzeichen von de
37EB  7C              ld    a,h
37EC  F2F437          jp    p,37F4       Zahlen mit gleichem Vorzeichen vergleichen
37EF  87              add   a,a
37F0  9F              sbc   a,a
37F1  D8              ret   c
37F2  3C              inc   a
37F3  C9              ret

37F4  BA              cp    d
37F5  20F9            jr    nz,37F0
37F7  7D              ld    a,l
37F8  93              sub   e
37F9  20F5            jr    nz,37F0
37FB  C9              ret
```

```
************************************************************ ROM-Header
C000  80            db    80          erstes Vordergrund-ROM
C001  01            db    01          Mark 1
C002  00            db    00          Version 0
C003  00            db    00          Modifikation 0
C004  4CC0          dw    C04C        Adresse des Namens

******************************************************** BASIC-Initialisierung
C006  3100C0        ld    sp,C000     Stack ab C000
C009  CDCBBC        call  BCCB        KL ROM WALK
C00C  CDC4F4        call  F4C4        Speicher konfigurieren
C00F  DA0000        jp    c,0000      zu wenig Speicher, dann Reset

C012  2100AC        ld    hl,AC00
C015  3600          ld    (hl),00
C017  061B          ld    b,1B
C019  23            inc   hl
C01A  36C9          ld    (hl),C9     'ret' von AC01 bis AC1B
C01C  10FB          djnz  C019
C01E  213FC0        ld    hl,C03F     Zeiger auf ' BASIC 1.0'
C021  CD37C3        call  C337        ausgeben
C024  AF            xor   a
C025  3200AC        ld    (AC00),a    Flag für Blanks unterdrücken löschen
C028  CDCBDD        call  DDCB        aktuelle Zeilenadresse auf Null
C02B  CD84CA        call  CA84        Fehlernummer löschen
C02E  CD97BD        call  BD97        RND Init
C031  CDD3C0        call  C0D3        AUTO-Modus löschen
C034  CD3EC1        call  C13E        NEW-Befehl
C037  11F000        ld    de,00F0     240
C03A  CD06F7        call  F706        SYMBOL AFTER 240
C03D  1825          jr    C064        zum READY-Modus

************************************************************************
C03F  20 42 41 53 49 43 20 31
C047  2E 30 0A 0A 00                  ' BASIC 1.0' LF,LF

************************************************************************
C04C  42 41 53 49 C3 00               'BASI', 'C'+80H, 00H

******************************************************** BASIC-Befehl EDIT
C052  CDE1CE        call  CEE1        Zeilennummer nach de holen
C055  C0            ret   nz
C056  3100C0        ld    sp,C000     Stack initialisieren
C059  CD9AE7        call  E79A        BASIC-Zeile de suchen (vorhanden?)
C05C  CD63E1        call  E163        BASIC-Zeile in Puffer listen
C05F  CD43CA        call  CA43        Eingabezeile holen
C062  3854          jr    c,C0B8

*********************************************************** READY-Modus
C064  CD01AC        call  AC01        ret
C067  3100C0        ld    sp,C000
C06A  CD62C1        call  C162
C06D  CDD6DD        call  DDD6        Zeilenadresse holen
C070  DCB6BC        call  c,BCB6      SOUND HOLD
C073  CD48BB        call  BB48        KM DISARM BREAK
```

```
C076  CD86C3         call   C386           Bildschirm initialisieren
C079  3A45AE         ld     a,(AE45)       geschütztes Programm ?
C07C  B7             or     a
C07D  C43EC1         call   nz,C13E        ja, Programm und Variablen löschen
C080  3AAAAD         ld     a,(ADAA)       ERROR-Nummer
C083  D602           sub    02             'Syntax error' ?
C085  2009           jr     nz,C090        nein
C087  32AAAD         ld     (ADAA),a       ERROR-Nummer auf Null
C08A  CDDFCA         call   CADF           Zeilennummer der ERROR-Zeile holen
C08D  EB             ex     de,hl
C08E  38C6           jr     c,C056         zum EDIT-Befehl
C090  21CCC0         ld     hl,C0CC        'Ready'
C093  CD41C3         call   C341           ausgeben
C096  CDCBDD         call   DDCB           aktuelle Zeilenadresse auf Null
C099  3A1CAC         ld     a,(AC1C)       AUTO-Flag gesetzt ?
C09C  B7             or     a
C09D  2811           jr     z,C0B0         nein
C09F  CD02C1         call   C102           nächste Zeilennummer vorgeben
C0A2  30C0           jr     nc,C064        zum READY-Modus
C0A4  7E             ld     a,(hl)
C0A5  B7             or     a
C0A6  28F1           jr     z,C099
C0A8  CDD2E6         call   E6D2           Statement in Interpreterkode wandeln
C0AB  CD7AC1         call   C17A
C0AE  18E9           jr     C099

*************************************************************************
C0B0  CD3BCA         call   CA3B           Eingabezeile holen
C0B3  30FB           jr     nc,C0B0        'ESC' gedrückt, dann wiederholen
C0B5  CD4EC3         call   C34E           LF ausgeben
C0B8  CDBCE6         call   E6BC           Zeile in Interpreterkode wandeln
C0BB  3005           jr     nc,C0C2        Direkt-Befehl ?
C0BD  C47AC1         call   nz,C17A
C0C0  18D4           jr     C096

C0C2  CDBBDE         call   DEBB           Zeile in Puffer ab &40 kopieren
C0C5  CD53C4         call   C453           Unterbrechung durch 'Break' erlauben
C0C8  2B             dec    hl
C0C9  C374DD         jp     DD74           zur Interpreterschleife

*************************************************************************
C0CC  52 65 61 64 79 0A 00                 'Ready', LF, 00H

************************************************************* AUTO-Modus löschen
C0D3  AF             xor    a              0
C0D4  1805           jr     C0DB

************************************************************* AUTO-Modus setzen
C0D6  221DAC         ld     (AC1D),hl      Zeilennummer
C0D9  3EFF           ld     a,FF
C0DB  321CAC         ld     (AC1C),a       Flag für AUTO setzen
C0DE  C9             ret
```

```
******************************************** BASIC-Befehl  AUTO
C0DF  110A00      ld    de,000A      10, Default
C0E2  2802        jr    z,C0E6
C0E4  FE2C        cp    2C           ','
C0E6  C4E1CE      call  nz,CEE1      Zeilennummer nach de holen
C0E9  D5          push  de
C0EA  110A00      ld    de,000A      10, Default
C0ED  CD55DD      call  DD55         folgt Komma ?
C0F0  DCE1CE      call  c,CEE1       ja, Zeilennummer nach de holen
C0F3  CD4ADD      call  DD4A         Zeilenende, sonst 'Syntax error'
C0F6  EB          ex    de,hl
C0F7  221FAC      ld    (AC1F),hl    AUTO-Inkrement merken
C0FA  E1          pop   hl
C0FB  CDD6C0      call  C0D6         Flag für AUTO-Modus setzen
C0FE  C1          pop   bc
C0FF  C396C0      jp    C096

C102  2A1DAC      ld    hl,(AC1D)    Zeilennummer
C105  E5          push  hl
C106  CD79EE      call  EE79         Zeilennummer ausgeben
C109  D1          pop   de
C10A  CDA3E7      call  E7A3         Zeile suchen
C10D  3E2A        ld    a,2A         '*'
C10F  3802        jr    c,C113       Zeile vorhanden ?
C111  3E20        ld    a,20         ' '
C113  CD56C3      call  C356         ausgeben
C116  CDD3C0      call  C0D3         AUTO-Modus löschen
C119  CD3BCA      call  CA3B         Eingabezeile holen
C11C  D0          ret   nc           ESC gedrückt ?
C11D  CD4EC3      call  C34E         LF ausgeben
C120  E5          push  hl
C121  2A1FAC      ld    hl,(AC1F)    Zeilennummer
C124  19          add   hl,de        plus Inkrement
C125  D4D6C0      call  nc,C0D6      AUTO-Modus setzen
C128  E1          pop   hl
C129  37          scf
C12A  C9          ret

******************************************** BASIC-Befehl  NEW
C12B  C0          ret   nz
C12C  CD3EC1      call  C13E         Programm und Variablen löschen
C12F  C364C0      jp    C064         zum READY-Modus

******************************************** BASIC-Befehl  CLEAR
C132  E5          push  hl
C133  CD8CC1      call  C18C
C136  CD5BC1      call  C15B
C139  CD7AC1      call  C17A
C13C  E1          pop   hl
C13D  C9          ret

**************************************** Programm und Variablen löschen
C13E  2A7FAE      ld    hl,(AE7F)    Beginn des freien RAMs
C141  EB          ex    de,hl
C142  2A7BAE      ld    hl,(AE7B)    HIMEM
```

```
C145 CDDAFF    call  FFDA       bc := hl - de
C148 62        ld    h,d
C149 6B        ld    l,e
C14A 13        inc   de
C14B AF        xor   a          Akku löschen
C14C 77        ld    (hl),a
C14D EDB0      ldir             Beginn freies RAM bis HIMEM löschen
C14F 3245AE    ld    (AE45),a   Flag für geschütztes Programm löschen
C152 CD76E6    call  E676       Programmende := Programmstart
C155 CD8CC1    call  C18C       Variablen löschen
C158 CD6BC1    call  C16B       s.u.
C15B CDADD2    call  D2AD       Kassetten-I/O abbrechen
C15E AF        xor   a
C15F CD73BD    call  BD73       RAD-Modus setzen
C162 CDB3FB    call  FBB3       Descriptorstack initialisieren
C165 CDFDD9    call  D9FD
C168 C39DC1    jp    C19D

C16B CDE6DD    call  DDE6       TROFF
C16E CDD3C0    call  C0D3       AUTO-Modus löschen
C171 CDF2F1    call  F1F2       TAB-Stops auf 13 setzen
C174 CD76E6    call  E676       Programmende := Programmstart
C177 CDB1D5    call  D5B1       Variablenzeiger rücksetzen
C17A CDD9CB    call  CBD9       ON-ERROR löschen
C17D CDABCB    call  CBAB       CONT sperren
C180 CDEDC8    call  C8ED       SOUND und Event-Reset
C183 CD8EF5    call  F58E       BASIC-Stack initialisieren
C186 CDD2D5    call  D5D2       Flag für FN löschen
C189 C3E5DC    jp    DCE5       RESTORE

*********************************************** Variablen löschen
C18C C5        push  bc
C18D E5        push  hl
C18E CDCAF5    call  F5CA       Stringzeiger rücksetzen
C191 CDAED5    call  D5AE       Variablenzeiger rücksetzen
C194 CDFCD5    call  D5FC       Variablen A-Z auf 'Real'
C197 CD89E9    call  E989
C19A E1        pop   hl
C19B C1        pop   bc
C19C C9        ret

*************************************************************
C19D AF        xor   a
C19E CDAFC1    call  C1AF
C1A1 AF        xor   a
C1A2 E5        push  hl
C1A3 F5        push  af
C1A4 FE08      cp    08         < 8 ?
C1A6 DCB4BB    call  c,BBB4     TXT STR SELECT
C1A9 F1        pop   af
C1AA 2121AC    ld    hl,AC21    aktuelle Streamnummer
C1AD 1804      jr    C1B3
```

```
C1AF  E5          push  hl
C1B0  2122AC      ld    hl,AC22       Eingabekanal
C1B3  D5          push  de
C1B4  5F          ld    e,a
C1B5  7E          ld    a,(hl)
C1B6  73          ld    (hl),e
C1B7  D1          pop   de
C1B8  E1          pop   hl
C1B9  C9          ret

C1BA  3A21AC      ld    a,(AC21)      aktuelle Streamnummer
C1BD  FE08        cp    08            Drucker ?
C1BF  C9          ret

C1C0  3A22AC      ld    a,(AC22)      Eingabekanal
C1C3  FE09        cp    09            Kassette ?
C1C5  C9          ret

C1C6  CDE3C1      call  C1E3
C1C9  18D7        jr    C1A2

C1CB  CDE3C1      call  C1E3
C1CE  18DF        jr    C1AF

******************************************** Streamnummer holen
C1D0  CDE3C1      call  C1E3
C1D3  FE08        cp    08
C1D5  302E        jr    nc,C205       'Improper argument'
C1D7  CDA2C1      call  C1A2
C1DA  C1          pop   bc
C1DB  F5          push  af
C1DC  CDF9FF      call  FFF9          jp (bc) Funktion ausführen
C1DF  F1          pop   af
C1E0  C3A2C1      jp    C1A2

******************************************** auf Streamnummer testen
C1E3  7E          ld    a,(hl)
C1E4  FE23        cp    23            '#'
C1E6  3E00        ld    a,00          0 als Default
C1E8  C0          ret   nz
C1E9  CDF5C1      call  C1F5          Streamnummer holen
C1EC  F5          push  af
C1ED  CD55DD      call  DD55          folgt Komma ?
C1F0  D44ADD      call  nc,DD4A       nein, dann Ende des Statements ?
C1F3  F1          pop   af
C1F4  C9          ret

******************************************** Streamnummer holen
C1F5  CD37DD      call  DD37          Test auf nachfolgendes Zeichen
C1F8  23          db    23            '#'
C1F9  3E0A        ld    a,0A          10, Maximalwert
C1FB  C5          push  bc
C1FC  D5          push  de
C1FD  47          ld    b,a
C1FE  CD67CE      call  CE67          8-Bit-Wert holen
```

```
C201  B8           cp    b          mit b vergleichen
C202  D1           pop   de
C203  C1           pop   bc
C204  D8           ret   c          kleiner b, ok
C205  1E05         ld    e,05       'Improper argument'
C207  C394CA       jp    CA94       Fehlermeldung ausgeben

******************************************** BASIC-Befehl PAPER
C20A  CDD0C1       call  C1D0       Streamnummer holen
C20D  0196BB       ld    bc,BB96    TXT SET PAPER
C210  1806         jr    C218

********************************************* BASIC-Befehl PEN
C212  CDD0C1       call  C1D0       Streamnummer holen
C215  0190BB       ld    bc,BB90    TXT SET PEN
C218  CD4BC2       call  C24B       Argument < 16 holen
C21B  E5           push  hl
C21C  CDF9FF       call  FFF9       jp (bc) Funktion ausführen
C21F  E1           pop   hl
C220  C9           ret

******************************************* BASIC-Befehl BORDER
C221  CD3CC2       call  C23C       Argument(e) < 32 holen
C224  E5           push  hl
C225  CD38BC       call  BC38       SCR SET BORDER
C228  E1           pop   hl
C229  C9           ret

********************************************* BASIC-Befehl INK
C22A  CD4BC2       call  C24B       Argument < 16 holen
C22D  F5           push  af
C22E  CD37DD       call  DD37       Test auf nachfolgendes Zeichen
C231  2C           db    2C         ','
C232  CD3CC2       call  C23C       Argument(e) < 32 holen
C235  F1           pop   af
C236  E5           push  hl
C237  CD32BC       call  BC32       SCR SET INK
C23A  E1           pop   hl
C23B  C9           ret

******************************************* Argument(e) < 32 holen
C23C  CD44C2       call  C244       Argument < 32 holen
C23F  41           ld    b,c
C240  CD55DD       call  DD55       folgt Komma ?
C243  D0           ret   nc         nein
C244  3E20         ld    a,20       32
C246  CDFBC1       call  C1FB       Argument < 32 holen
C249  4F           ld    c,a
C24A  C9           ret

******************************************** Argument < 16 holen
C24B  3E10         ld    a,10       16
C24D  18AC         jr    C1FB       Argument < 16 holen
```

```
*********************************************** BASIC-Befehl MODE
C24F  3E03       ld     a,03        3
C251  CDFBC1     call   C1FB        Argument < 3 holen
C254  E5         push   hl
C255  CD0EBC     call   BC0E        SCR SET MODE
C258  E1         pop    hl
C259  C9         ret

************************************************ BASIC-Befehl CLS
C25A  CDD0C1     call   C1D0        Streamnummer holen
C25D  3E0C       ld     a,0C        FF
C25F  C36EC3     jp     C36E        ausgeben

****************************************************** VPOS
C262  0167C2     ld     bc,C267
C265  1812       jr     C279

C267  3A21AC     ld     a,AC21      aktuelle Streamnummer
C26A  FE08       cp     08          > 8 ?
C26C  3097       jr     nc,C205     'Improper argument'
C26E  CD78BB     call   BB78        TXT GET CURSOR
C271  CD87BB     call   BB87        TXT VALIDATE
C274  7D         ld     a,l
C275  C9         ret

****************************************************** POS
C276  0190C2     ld     bc,C290
C279  CDF5C1     call   C1F5        Wert < 10 holen
C27C  CDA2C1     call   C1A2        Select Stream
C27F  F5         push   af
C280  CD37DD     call   DD37        Test auf nachfolgendes Zeichen
C283  29         db     29          ')'
C284  E5         push   hl
C285  CDF9FF     call   FFF9        jp (bc) Funktion ausführen
C288  CD0AFF     call   FF0A        Akkuinhalt als Integerzahl übernehmen
C28B  E1         pop    hl
C28C  F1         pop    af
C28D  C3A2C1     jp     C1A2        Select Stream

********************************************* aktuelle PRINT-Position holen
C290  3A21AC     ld     a,(AC21)    aktuelle Streamnummer
C293  FE08       cp     08
C295  CADFC3     jp     z,C3DF      Drucker-Position holen
C298  3A25AC     ld     a,(AC25)    Kassetten-Position holen
C29B  D0         ret    nc
C29C  C39CC3     jp     C39C        Bildschirm-Position holen

***********************************************************
C29F  3A21AC     ld     a,(AC21)    aktuelle Streamnummer
C2A2  FE08       cp     08          Drucker ?
C2A4  280D       jr     z,C2B3      ja
C2A6  D0         ret    nc          Kassette ?
C2A7  D5         push   de
C2A8  E5         push   hl
C2A9  CD69BB     call   BB69        TXT GET WINDOW
```

```
C2AC  7A           ld     a,d
C2AD  94           sub    h
C2AE  3C           inc    a
C2AF  E1           pop    hl
C2B0  D1           pop    de
C2B1  37           scf
C2B2  C9           ret

C2B3  3A24AC       ld     a,(AC24)        WIDTH
C2B6  FEFF         cp     FF
C2B8  C9           ret

C2B9  E5           push   hl
C2BA  CDBFC2       call   C2BF
C2BD  E1           pop    hl
C2BE  C9           ret

C2BF  67           ld     h,a
C2C0  CD9FC2       call   C29F
C2C3  3F           ccf
C2C4  D8           ret    c
C2C5  6F           ld     l,a
C2C6  CD90C2       call   C290
C2C9  3D           dec    a
C2CA  37           scf
C2CB  C8           ret    z
C2CC  84           add    a,h
C2CD  3F           ccf
C2CE  D0           ret    nc
C2CF  3D           dec    a
C2D0  BD           cp     l
C2D1  C9           ret

**************************************** BASIC-Befehl LOCATE
C2D2  CDD0C1       call   C1D0            Streamnummer holen
C2D5  CD27C3       call   C327            2 8-Bit-Werte ungleich Null holen
C2D8  E5           push   hl
C2D9  EB           ex     de,hl
C2DA  24           inc    h
C2DB  2C           inc    l
C2DC  CD75BB       call   BB75            TXT SET CUSROR
C2DF  E1           pop    hl
C2E0  C9           ret

**************************************** BASIC-Befehl WINDOW
C2E1  7E           ld     a,(hl)
C2E2  FEE7         cp     E7              'SWAP'
C2E4  2817         jr     z,C2FD
C2E6  CDD0C1       call   C1D0            Streamnummer holen
C2E9  CD27C3       call   C327            2 8-Bit-Werte ungleich Null holen
C2EC  D5           push   de
C2ED  CD37DD       call   DD37            Test auf nachfolgendes Zeichen
C2F0  2C           db     2C              ','
C2F1  CD27C3       call   C327            2 8-Bit-Werte ungleich Null holen
C2F4  E3           ex     (sp),hl
```

```
C2F5  7A              ld     a,d
C2F6  55              ld     d,l
C2F7  6F              ld     l,a
C2F8  CD66BB          call   BB66          TXT WIN ENABLE
C2FB  E1              pop    hl
C2FC  C9              ret

*********************************************** WINDOW SWAP
C2FD  CD3FDD          call   DD3F          Blanks überlesen
C300  CD12C3          call   C312          Argument < 8 holen
C303  48              ld     c,b
C304  CD55DD          call   DD55          folgt Komma ?
C307  0600            ld     b,00          Defaultwert 0
C309  DC12C3          call   c,C312        ja, Argument < 8 holen
C30C  E5              push   hl
C30D  CDB7BB          call   BBB7          TXT SWAP STREAMS
C310  E1              pop    hl
C311  C9              ret

********************************************** Argument < 8 holen
C312  3E08            ld     a,08          8
C314  CDFBC1          call   C1FB          Argument kleiner 8 holen
C317  47              ld     b,a
C318  C9              ret

********************************************** BASIC-Befehl TAG
C319  CDD0C1          call   C1D0          Streamnummer holen
C31C  3EFF            ld     a,FF
C31E  1804            jr     C324

******************************************** BASIC-Befehl TAGOFF
C320  CDD0C1          call   C1D0          Streamnummer holen
C323  AF              xor    a
C324  C363BB          jp     BB63          TXT SET GRAPHIC

************************************** 2 8-Bit-Werte ungleich Null holen
C327  CD2FC3          call   C32F          ersten Wert holen
C32A  53              ld     d,e
C32B  CD37DD          call   DD37          Test auf nachfolgendes Zeichen
C32E  2C              db     2C            ','
C32F  D5              push   de
C330  CD6DCE          call   CE6D          8-Bit-Wert ungleich Null holen
C333  D1              pop    de
C334  5F              ld     e,a
C335  1D              dec    e
C336  C9              ret

*********************************************** String ausgeben
C337  3E84            ld     a,84          132
C339  3224AC          ld     (AC24),a      WIDTH auf 132
C33C  E5              push   hl            Startadresse des Strings
C33D  CD9DC1          call   C19D          Ausgabekanal selektieren
C340  E1              pop    hl
C341  F5              push   af
C342  E5              push   hl
```

```
C343  7E           ld    a,(hl)     Zeichen holen
C344  23           inc   hl         Zeiger erhöhen
C345  B7           or    a          Nullbyte gleich Ende des Strings
C346  C456C3       call  nz,C356    Zeichen ausgeben
C349  20F8         jr    nz,C343    nicht null, dann nächstes Zeichen
C34B  E1           pop   hl
C34C  F1           pop   af
C34D  C9           ret

************************************************** LF ausgeben
C34E  F5           push  af
C34F  3E0A         ld    a,0A       LF
C351  CD56C3       call  C356       ausgeben
C354  F1           pop   af
C355  C9           ret

************************************************* Zeichen ausgeben
C356  F5           push  af
C357  CD5CC3       call  C35C       Zeichen ausgeben
C35A  F1           pop   af
C35B  C9           ret

C35C  FE0A         cp    0A         LF ?
C35E  200E         jr    nz,C36E
C360  3A21AC       ld    a,(AC21)   aktuelle Streamnummer
C363  FE08         cp    08         Drucker ?
C365  CAA8C3       jp    z,C3A8     ja
C368  D2EAC3       jp    nc,C3EA    Kassette
C36B  C392C3       jp    C392       Bildschirm

************************************************* Zeichen ausgeben
C36E  F5           push  af
C36F  C5           push  bc
C370  4F           ld    c,a        Zeichen nach c
C371  CD77C3       call  C377       ausgeben
C374  C1           pop   bc
C375  F1           pop   af
C376  C9           ret

********************************************* Ausgabestrom selektieren
C377  3A21AC       ld    a,(AC21)   aktuelle Streamnummer
C37A  FE08         cp    08
C37C  CAB5C3       jp    z,C3B5     Drucker ?
C37F  D2F8C3       jp    nc,C3F8    Kassette ?
C382  79           ld    a,c        Zeichen in Akku
C383  C399C3       jp    C399       Zeichen auf Bildschirm ausgeben

********************************************** Bildschirm initialisieren
C386  AF           xor   a
C387  CD63BB       call  BB63       TXT SET GRAPHIC

C38A  CD54BB       call  BB54       TXT VDU ENABLE
C38D  CD9CC3       call  C39C       Cursor auf zulässige Position
C390  3D           dec   a
C391  C8           ret   z
C392  3E0D         ld    a,0D       CR
```

```
C394  CD99C3        call   C399        ausgeben
C397  3E0A          ld     a,0A        LF
C399  C35ABB        jp     BB5A        TXT OUTPUT

******************************************** Cursor auf zulässige Position
C39C  C5            push   bc
C39D  E5            push   hl
C39E  CD78BB        call   BB78        TXT GET CURSOR
C3A1  CD87BB        call   BB87        TXT VALIDATE
C3A4  7C            ld     a,h
C3A5  E1            pop    hl
C3A6  C1            pop    bc
C3A7  C9            ret

******************************************** CR & LF auf Drucker ausgeben
C3A8  C5            push   bc
C3A9  0E0D          ld     c,0D        CR
C3AB  CDB5C3        call   C3B5        ausgeben
C3AE  0E0A          ld     c,0A        LF
C3B0  CDB5C3        call   C3B5        ausgeben
C3B3  C1            pop    bc
C3B4  C9            ret

******************************************** Zeichen auf Drucker ausgeben
C3B5  E5            push   hl
C3B6  79            ld     a,c
C3B7  EE0D          xor    0D          CR
C3B9  2813          jr     z,C3CE
C3BB  79            ld     a,c
C3BC  FE20          cp     20          ' '
C3BE  3814          jr     c,C3D4      Kontroll-Zeichen nicht zählen
C3C0  2A23AC        ld     hl,(AC23)   aktuelle Drucker-Position und WIDTH
C3C3  24            inc    h
C3C4  7D            ld     a,l
C3C5  2807          jr     z,C3CE
C3C7  BC            cp     h
C3C8  CCA8C3        call   z,C3A8
C3CB  3A23AC        ld     a,(AC23)    aktuelle Drucker-Position
C3CE  3C            inc    a
C3CF  2803          jr     z,C3D4
C3D1  3223AC        ld     (AC23),a    aktuelle Drucker-Position
C3D4  E1            pop    hl
C3D5  79            ld     a,c
C3D6  CD2BBD        call   BD2B        MC PRINT CHAR
C3D9  D8            ret    c           Ausgabe ok ?
C3DA  CD3CC4        call   C43C        Unterbrechung durch 'ESC' ?
C3DD  18F6          jr     C3D5

*****************************************************************
C3DF  3A23AC        ld     a,(AC23)    aktuelle Drucker-Position
C3E2  C9            ret
```

```
********************************************** BASIC-Befehl WIDTH
C3E3 CD6DCE       call  CE6D         8-Bit-Wert ungleich Null holen
C3E6 3224AC       ld    (AC24),a     WIDTH setzen
C3E9 C9           ret

********************************************* neue Zeile auf Kassette
C3EA 3E01         ld    a,01
C3EC 3225AC       ld    (AC25),a     Kassettenposition auf eins
C3EF 3E0D         ld    a,0D         CR
C3F1 CD0DC4       call  C40D         auf Kassette ausgeben
C3F4 3E0A         ld    a,0A         LF
C3F6 1815         jr    C40D         auf Kassette ausgeben

*******************************************************************
C3F8 E5           push  hl
C3F9 2125AC       ld    hl,AC25      Kassettenposition
C3FC 79           ld    a,c
C3FD 0601         ld    b,01         bei neuer Zeile Position auf eins
C3FF FE0D         cp    0D           CR
C401 2808         jr    z,C40B
C403 FE20         cp    20           ' '
C405 3805         jr    c,C40C       Kontrollzeichen nicht zählen
C407 46           ld    b,(hl)       Zeichenzähler laden
C408 04           inc   b            und erhöhen
C409 2801         jr    z,C40C
C40B 70           ld    (hl),b       neuen Zählerwert merken
C40C E1           pop   hl
C40D CD95BC       call  BC95         CAS OUT CHAR
C410 D8           ret   c            nicht ESC-Taste gedrückt ?
C411 C36BCB       jp    CB6B         'Break', READY-Modus

*******************************************************************
C414 C386BC       jp    BC86         CAS RETURN

********************************************* reservierte Variable EOF
C417 E5           push  hl
C418 CD89BC       call  BC89         CAS TEST EOF
C41B 28F4         jr    z,C411       ESC gedrückt ?
C41D 3F           ccf
C41E 9F           sbc   a,a
C41F CD05FF       call  FF05         Vorzeichen als Integerzahl übernehmen
C422 E1           pop   hl
C423 C9           ret

************************************** Ein Zeichen vom Eingabekanal holen
C424 3A22AC       ld    a,(AC22)     Eingabekanal
C427 FE09         cp    09           Kassette ?
C429 CA80BC       jp    z,BC80       ja, CAS IN CHAR
C42C CD09BB       call  BB09         KM READ CHAR
```

```
C42F  D8          ret    c          Taste gedrückt ?
C430  CD81BB      call   BB81       TXT CUR ON
C433  CD06BB      call   BB06       KM WAIT CHAR
C436  C384BB      jp     BB84       TXT CUR OFF

*******************************************************
C439  C309BB      jp     BB09       KM READ CHAR

******************************************** Test auf Abbruch mit 'ESC'
C43C  CD09BB      call   BB09       KM READ CHAR
C43F  D0          ret    nc
C440  FEFC        cp     FC         'Break' ?
C442  C0          ret    nz
C443  C5          push   bc
C444  D5          push   de
C445  E5          push   hl
C446  CD6FC4      call   C46F       auf zweiten Tastendruck warten
C449  DA6BCB      jp     c,CB6B     'ESC', dann Abbruch
C44C  CD53C4      call   C453       Unterbrechung durch 'Break' erlauben
C44F  E1          pop    hl
C450  D1          pop    de
C451  C1          pop    bc
C452  C9          ret

************************************** Unterbrechung durch 'Break' erlauben
C453  E5          push   hl
C454  115EC4      ld     de,C45E    Adresse der Break-Event-Routine
C457  0EFD        ld     c,FD       BASIC-ROM selektiert
C459  CD45BB      call   BB45       KM ARM BREAK
C45C  E1          pop    hl
C45D  C9          ret

*********************************************** Break-Event Routine
C45E  E5          push   hl
C45F  CD09BB      call   BB09       KM READ CHAR
C462  3004        jr     nc,C468    keine Taste gedrückt ?
C464  FEEF        cp     EF         Break durch 'ESC' ?
C466  20F7        jr     nz,C45F    Tastendrücke vor 'ESC' ignorieren
C468  CD6FC4      call   C46F       warten auf zweites 'ESC'
C46B  E1          pop    hl
C46C  C347C8      jp     C847       Test auf ON BREAK GOSUB

************************************** Warten auf Tastendruck nach 'ESC'
C46F  CDB6BC      call   BCB6       SOUND HOLD
C472  F5          push   af
C473  CD30C4      call   C430       auf einen Tastendruck warten
C476  FEEF        cp     EF         Break durch 'ESC' ?
C478  28F9        jr     z,C473
C47A  FEFC        cp     FC         'Break' ?
C47C  280B        jr     z,C489
C47E  FE20        cp     20         ' ' ?
```

```
C480  C40CBB     call   nz,BB0C       nein, Zeichen merken KM CHAR RETURN
C483  F1         pop    af
C484  DCB9BC     call   c,BCB9        SOUND CONTINUE
C487  B7         or     a
C488  C9         ret



C489  F1         pop    af
C48A  37         scf
C48B  C9         ret

****************************************** BASIC-Befehl ORIGIN
C48C  CD1AC5     call   C51A          2 Argumente holen
C48F  C5         push   bc
C490  D5         push   de
C491  CD55DD     call   DD55          folgt Komma ?
C494  3018       jr     nc,C4AE       nein
C496  CD1AC5     call   C51A          2 Argumente holen
C499  C5         push   bc
C49A  D5         push   de
C49B  CD37DD     call   DD37          Test auf nachfolgendes Zeichen
C49E  2C         db     2C            ','
C49F  CD1AC5     call   C51A          2 Argumente holen
C4A2  C5         push   bc
C4A3  E3         ex     (sp),hl
C4A4  CDD2BB     call   BBD2          GRA WIN HEIGHT
C4A7  E1         pop    hl
C4A8  D1         pop    de
C4A9  E3         ex     (sp),hl
C4AA  CDCFBB     call   BBCF          GRA WIN WIDTH
C4AD  E1         pop    hl
C4AE  D1         pop    de
C4AF  E3         ex     (sp),hl
C4B0  CDC9BB     call   BBC9          GRA SET ORIGIN
C4B3  E1         pop    hl
C4B4  C9         ret

C4B5  CD51DD     call   DD51          Ende des Statements ?
C4B8  3806       jr     c,C4C0        ja
C4BA  CD4BC2     call   C24B          Argument < 16 holen
C4BD  CDE4BB     call   BBE4          GRA SET PAPER
C4C0  E5         push   hl
C4C1  CDDBBB     call   BBDB          GRA CLEAR WINDOW
C4C4  E1         pop    hl
C4C5  C9         ret

****************************************** BASIC-Befehl DRAW
C4C6  01F6BB     ld     bc,BBF6       GRA LINE ABSOLUTE
C4C9  180D       jr     C4D8
```

```
********************************************* BASIC-Befehl DRAWR
C4CB 01F9BB        ld     bc,BBF9       GRA LINE RELATIVE
C4CE 1808          jr     C4D8

********************************************** BASIC-Befehl PLOT
C4D0 01EABB        ld     bc,BBEA       GRA PLOT ABSOLUTE
C4D3 1803          jr     C4D8

********************************************** BASIC-Befehl PLOTR
C4D5 01EDBB        ld     bc,BBED       GRA PLOT RELATIVE
C4D8 C5            push   bc
C4D9 CD1AC5        call   C51A          2 Argumente holen
C4DC CD55DD        call   DD55          folgt Komma ?
C4DF 3006          jr     nc,C4E7       nein
C4E1 CD4BC2        call   C24B          Argument < 16 holen
C4E4 CDDEBB        call   BBDE          GRA SET PEN
C4E7 1828          jr     C511

***************************************************** TEST
C4E9 01F0BB        ld     bc,BBF0       GRA TEST ABSOLUTE
C4EC 1803          jr     C4F1

**************************************************** TESTR
C4EE 01F3BB        ld     bc,BBF3       GRA TEST RELATIVE
C4F1 C5            push   bc
C4F2 CD1AC5        call   C51A          2 Argumente holen
C4F5 CD37DD        call   DD37          Test auf nachfolgendes Zeichen
C4F8 29            db     29            ')'
C4F9 E3            ex     (sp),hl
C4FA C5            push   bc
C4FB E3            ex     (sp),hl
C4FC C1            pop    bc
C4FD CDF9FF        call   FFF9          jp (bc), Funktion ausführen
C500 CD0AFF        call   FF0A          Akkuinhalt als Integerzahl übernehmen
C503 E1            pop    hl
C504 C9            ret

********************************************** BASIC-Befehl MOVE
C505 01C0BB        ld     bc,BBC0       GRA MOVE ABSOLUTE
C508 1803          jr     C50D

********************************************** BASIC-Befehl MOVER
C50A 01C3BB        ld     bc,BBC3       GRA MOVE RELATIVE
C50D C5            push   bc
C50E CD1AC5        call   C51A          2 Argumente holen
C511 E3            ex     (sp),hl
C512 C5            push   bc
C513 E3            ex     (sp),hl
C514 C1            pop    bc
C515 CDF9FF        call   FFF9          jp (bc), Funktion ausführen
C518 E1            pop    hl
C519 C9            ret
```

```
************************************* 2 Integerargumente nach de, bc holen
C51A  CD86CE    call   CE86        16-Bit-Wert -32768 - +32767 holen
C51D  D5        push   de
C51E  CD37DD    call   DD37        Test auf nachfolgendes Zeichen
C521  2C        db     2C          ','
C522  CD86CE    call   CE86        16-Bit-Wert -32768 - +32767 holen
C525  42        ld     b,d
C526  4B        ld     c,e         2. Argument nach bc
C527  D1        pop    de          1. Argument
C528  C9        ret


******************************************************* BASIC-Befehl FOR
C529  CDB3D6    call   D6B3        Variable lesen
C52C  E5        push   hl
C52D  C5        push   bc
C52E  D5        push   de
C52F  CDC5C9    call   C9C5        zugehöriges NEXT suchen
C532  222CAC    ld     (AC2C),hl   Adresse merken
C535  D5        push   de
C536  E5        push   hl
C537  EB        ex     de,hl
C538  CD32C6    call   C632        offene FOR-NEXT-Schleife suchen
C53B  CCACF5    call   z,F5AC      gefunden, BASIC-Stackpointer setzen
C53E  E1        pop    hl
C53F  CD51DD    call   DD51        Ende des Statements ?
C542  110000    ld     de,0000     Default Null
C545  D486D6    call   nc,D686     nein, Variable holen
C548  44        ld     b,h
C549  4D        ld     c,l
C54A  E1        pop    hl
C54B  E3        ex     (sp),hl
C54C  7A        ld     a,d
C54D  B3        or     e
C54E  C4B8FF    call   nz,FFB8     Vergleich hl <> de
C551  C2F6C5    jp     nz,C5F6     'Unexpected NEXT'
C554  EB        ex     de,hl
C555  CDD2DD    call   DDD2        aktuelle Zeilenadresse nach hl
C558  E3        ex     (sp),hl
C559  CDCEDD    call   DDCE        aktuelle Zeilenadresse setzen
C55C  E1        pop    hl
C55D  F1        pop    af
C55E  E3        ex     (sp),hl
C55F  D5        push   de
C560  C5        push   bc
C561  E5        push   hl
C562  010516    ld     bc,1605     22 Bytes, Typ 5 'Real'
C565  B9        cp     c
C566  280B      jr     z,C573
```

```
C568  010210    ld    bc,1002     16 Bytes, Typ 2 'Integer'
C56B  B9        cp    c
C56C  2805      jr    z,C573
C56E  1E0D      ld    e,0D        'Type mismatch'
C570  C394CA    jp    CA94        Fehlermeldung ausgeben

C573  78        ld    a,b
C574  CDB0F5    call  F5B0        Platz im BASIC-Stack reservieren
C577  73        ld    (hl),e
C578  23        inc   hl          Variablenadresse auf BASIC-Stack
C579  72        ld    (hl),d
C57A  23        inc   hl
C57B  E3        ex    (sp),hl
C57C  CD37DD    call  DD37        Test auf nachfolgendes Zeichen
C57F  EF        db    EF          '='
C580  CDFBCE    call  CEFB        Ausdruck holen
C583  79        ld    a,c
C584  CDD7FE    call  FED7        Variablentyp vergleichen
C587  E5        push  hl
C588  2127AC    ld    hl,AC27     Zwischenspeicher für FOR-Variable
C58B  CD62FF    call  FF62        Variable nach hl kopieren
C58E  E1        pop   hl
C58F  CD37DD    call  DD37        Test auf nachfolgendes Zeichen
C592  EC        db    EC          'TO'
C593  CDFBCE    call  CEFB        Ausdruck holen
C596  E3        ex    (sp),hl
C597  79        ld    a,c
C598  CDD7FE    call  FED7        Variablentyp vergleichen
C59B  CD62FF    call  FF62        Endwert auf BASIC-Stack
C59E  EB        ex    de,hl
C59F  E3        ex    (sp),hl
C5A0  EB        ex    de,hl
C5A1  210100    ld    hl,0001     eins als Default STEP-Wert
C5A4  CD0DFF    call  FF0D        Integerzahl hl übernehmen
C5A7  EB        ex    de,hl
C5A8  7E        ld    a,(hl)
C5A9  FEE6      cp    E6          'STEP'
C5AB  2006      jr    nz,C5B3
C5AD  CD3FDD    call  DD3F        Blanks überlesen
C5B0  CDFBCE    call  CEFB        Ausdruck holen
C5B3  79        ld    a,c
C5B4  CDD7FE    call  FED7        Variablentyp vergleichen
C5B7  E3        ex    (sp),hl
C5B8  CD62FF    call  FF62        Variable nach (hl) kopieren
C5BB  CDA3FD    call  FDA3        Vorzeichen holen
C5BE  EB        ex    de,hl
C5BF  77        ld    (hl),a      Vorzeichen von STEP auf BASIC-Stack
C5C0  23        inc   hl
C5C1  EB        ex    de,hl
C5C2  E1        pop   hl
C5C3  CD4ADD    call  DD4A        Ende des Statements, sonst 'Syntax error'
C5C6  EB        ex    de,hl
```

```
C5C7  73            ld     (hl),e
C5C8  23            inc    hl             Adresse des FOR-Befehls auf BASIC-Stack
C5C9  72            ld     (hl),d
C5CA  23            inc    hl
C5CB  EB            ex     de,hl
C5CC  CDD2DD        call   DDD2           aktuelle Zeilenadresse nach hl
C5CF  EB            ex     de,hl
C5D0  73            ld     (hl),e
C5D1  23            inc    hl             Zeilenadresse von FOR auf BASIC-Stack
C5D2  72            ld     (hl),d
C5D3  23            inc    hl
C5D4  D1            pop    de
C5D5  73            ld     (hl),e
C5D6  23            inc    hl             Adresse des NEXT-Befehls auf BASIC-Stack
C5D7  72            ld     (hl),d
C5D8  23            inc    hl
C5D9  ED5B2CAC      ld     de,(AC2C)
C5DD  73            ld     (hl),e
C5DE  23            inc    hl             Zeilenadresse NEXT-Befehl auf BASIC-Stack
C5DF  72            ld     (hl),d
C5E0  23            inc    hl
C5E1  70            ld     (hl),b         &10 oder &16 für Integer/Real auf Stack
C5E2  D1            pop    de
C5E3  2127AC        ld     hl,AC27        Zeiger auf Zwischenspeicher
C5E6  CD66FF        call   FF66           FOR-Variable zurückholen
C5E9  AF            xor    a
C5EA  3226AC        ld     (AC26),a       Flag für ersten Durchlauf
C5ED  E1            pop    hl
C5EE  CDCEDD        call   DDCE           aktuelle Zeilenadresse setzen
C5F1  2A2CAC        ld     hl,(AC2C)
C5F4  180A          jr     C600           zum NEXT-Befehl

C5F6  1E01          ld     e,01           'Unexpected NEXT'
C5F8  C394CA        jp     CA94           Fehlermeldung ausgeben

******************************************* BASIC-Befehl NEXT
C5FB  3EFF          ld     a,FF
C5FD  3226AC        ld     (AC26),a       Flag für Inkrement addieren
C600  EB            ex     de,hl
C601  CD32C6        call   C632           offene FOR-Next-Schleife suchen
C604  20F0          jr     nz,C5F6        'Unexpected NEXT'
C606  EB            ex     de,hl
C607  CDACF5        call   F5AC           BASIC-Stackpointer setzen
C60A  EB            ex     de,hl
C60B  E5            push   hl
C60C  CD61C6        call   C661           Test auf Schleifenende
C60F  280F          jr     z,C620
C611  F1            pop    af
C612  23            inc    hl
C613  5E            ld     e,(hl)
C614  23            inc    hl             Programmzeiger nach de
C615  56            ld     d,(hl)
```

```
C616  23        inc   hl
C617  7E        ld    a,(hl)
C618  23        inc   hl          Zeilenadresse nach hl
C619  66        ld    h,(hl)
C61A  6F        ld    l,a
C61B  CDCEDD    call  DDCE        aktuelle Zeilenadresse setzen
C61E  EB        ex    de,hl
C61F  C9        ret

C620  010500    ld    bc,0005     BASIC-Stackpointer
C623  09        add   hl,bc       plus 5
C624  5E        ld    e,(hl)
C625  23        inc   hl          Programmzeiger nach 'NEXT'
C626  56        ld    d,(hl)
C627  E1        pop   hl
C628  CDACF5    call  F5AC        BASIC-Stackpointer setzen
C62B  EB        ex    de,hl
C62C  CD55DD    call  DD55        folgt Komma ?
C62F  38CF      jr    c,C600      ja, nächste NEXT-Schleife
C631  C9        ret
```

```
*************************************** offene FOR-NEXT-Schleife suchen
C632  2A8BB0     ld    hl,(B08B)      BASIC-Stackpointer
C635  E5         push  hl
C636  2B         dec   hl
C637  46         ld    b,(hl)
C638  23         inc   hl
C639  7D         ld    a,l
C63A  90         sub   b
C63B  6F         ld    l,a
C63C  9F         sbc   a,a
C63D  84         add   a,h
C63E  67         ld    h,a
C63F  E3         ex    (sp),hl
C640  78         ld    a,b
C641  FE07       cp    07             'WHILE-WEND' ?
C643  2819       jr    z,C65E
C645  FE10       cp    10             Integer 'FOR-NEXT' ?
C647  2804       jr    z,C64D
C649  FE16       cp    16             Real 'FOR-NEXT' ?
C64B  200D       jr    nz,C65A
C64D  E5         push  hl
C64E  2B         dec   hl
C64F  2B         dec   hl
C650  7E         ld    a,(hl)
C651  2B         dec   hl
C652  6E         ld    l,(hl)
C653  67         ld    h,a
C654  CDB8FF     call  FFB8           Vergleich hl <> de
C657  E1         pop   hl
C658  2004       jr    nz,C65E
C65A  EB         ex    de,hl
C65B  E1         pop   hl
C65C  78         ld    a,b
C65D  C9         ret

C65E  E1         pop   hl
C65F  18D4       jr    C635

C661  5E         ld    e,(hl)
C662  23         inc   hl
C663  56         ld    d,(hl)
C664  23         inc   hl
C665  FE10       cp    10             Integer ?
C667  282D       jr    z,C696
C669  E5         push  hl
C66A  010500     ld    bc,0005        Typ auf 'Real'
C66D  79         ld    a,c
C66E  EB         ex    de,hl
C66F  CD4BFF     call  FF4B           Variable und Typ übernehmen
C672  E1         pop   hl
C673  3A26AC     ld    a,(AC26)       Flag für ersten Durchlauf
C676  B7         or    a
C677  2810       jr    z,C689         ja, Addition überspringen
C679  E5         push  hl
C67A  09         add   hl,bc
```

```
C67B  CDCCFC     call  FCCC       STEP-Wert addieren
C67E  E1         pop   hl
C67F  E5         push  hl
C680  2B         dec   hl
C681  56         ld    d,(hl)
C682  2B         dec   hl
C683  5E         ld    e,(hl)
C684  EB         ex    de,hl
C685  CD62FF     call  FF62       Variable nach (hl) kopieren
C688  E1         pop   hl
C689  E5         push  hl
C68A  0E05       ld    c,05
C68C  CD09FD     call  FD09       arithmetischer Vergleich
C68F  E1         pop   hl
C690  010A00     ld    bc,000A    10
C693  09         add   hl,bc
C694  96         sub   (hl)
C695  C9         ret

C696  E5         push  hl
C697  EB         ex    de,hl
C698  5E         ld    e,(hl)
C699  23         inc   hl
C69A  56         ld    d,(hl)
C69B  3A26AC     ld    a,(AC26)   erster Durchlauf ?
C69E  B7         or    a
C69F  2816       jr    z,C6B7     ja, Addition überspringen
C6A1  E3         ex    (sp),hl
C6A2  E5         push  hl
C6A3  23         inc   hl
C6A4  23         inc   hl
C6A5  7E         ld    a,(hl)
C6A6  23         inc   hl         STEP-Wert nach hl holen
C6A7  66         ld    h,(hl)
C6A8  6F         ld    l,a
C6A9  CDACBD     call  BDAC       Integer-Addition hl := hl + de
C6AC  1E06       ld    e,06       'Overflow'
C6AE  D294CA     jp    nc,CA94    Fehlermeldung ausgeben

C6B1  EB         ex    de,hl
C6B2  E1         pop   hl
C6B3  E3         ex    (sp),hl
C6B4  72         ld    (hl),d
C6B5  2B         dec   hl
C6B6  73         ld    (hl),e
C6B7  E1         pop   hl
C6B8  7E         ld    a,(hl)
C6B9  23         inc   hl
C6BA  E5         push  hl
C6BB  66         ld    h,(hl)
C6BC  6F         ld    l,a
C6BD  EB         ex    de,hl
C6BE  CDC4BD     call  BDC4       Integer-Vergleich
C6C1  E1         pop   hl
C6C2  23         inc   hl
```

```
C6C3 23          inc   hl
C6C4 23          inc   hl
C6C5 96          sub   (hl)
C6C6 C9          ret

****************************************************** BASIC-Befehl IF
C6C7 CDFBCE      call  CEFB        Ausdruck holen
C6CA FEA0        cp    A0          'GOTO'
C6CC 2804        jr    z,C6D2
C6CE CD37DD      call  DD37        Test auf nachfolgendes Zeichen
C6D1 EB          db    EB          'THEN'
C6D2 E5          push  hl
C6D3 CDA3FD      call  FDA3        Vorzeichen holen
C6D6 E1          pop   hl
C6D7 CC9FE8      call  z,E89F      Ende der Zeile oder ELSE-Zweig suchen
C6DA C8          ret   z
C6DB CD51DD      call  DD51        Ende des Statements ?
C6DE D8          ret   c           ja
C6DF FE1E        cp    1E          Zeilennummer ?
C6E1 2805        jr    z,C6E8      ja, zum GOTO-Befehl
C6E3 FE1D        cp    1D          Zeilenadresse ?
C6E5 C2ABDD      jp    nz,DDAB     nein,BASIC-Befehl ausführen

**************************************************** BASIC-Befehl GOTO
C6E8 CD67E7      call  E767        Zeilenadresse holen
C6EB EB          ex    de,hl       Adresse als Programmzeiger übernehmen
C6EC C9          ret

*************************************************** BASIC-Befehl GOSUB
C6ED CD67E7      call  E767        Zeilenadresse holen
C6F0 CDEFE8      call  E8EF        DATA-Befehl, Rest des Statements überlesen
C6F3 EB          ex    de,hl
C6F4 0E00        ld    c,00        Kennzeichen für normales 'GOSUB'
C6F6 E5          push  hl          Adresse des Unterprogramms merken
C6F7 3E06        ld    a,06        6 Bytes
C6F9 CDB0F5      call  F5B0        Platz im BASIC-Stack reservieren
C6FC 71          ld    (hl),c      Null
C6FD 23          inc   hl
C6FE 73          ld    (hl),e
C6FF 23          inc   hl          Adresse der Anweisung nach 'GOSUB'
C700 72          ld    (hl),d      auf BASIC-Stack
C701 23          inc   hl
C702 EB          ex    de,hl
C703 CDD2DD      call  DDD2        aktuelle Zeilenadresse nach hl
C706 EB          ex    de,hl
C707 73          ld    (hl),e
C708 23          inc   hl          Zeilenadresse auf BASIC-Stack
C709 72          ld    (hl),d
C70A 23          inc   hl
C70B 3606        ld    (hl),06     Kennzeichen für 'GOSUB'
C70D E1          pop   hl          Programmzeiger auf Unterprogramm
C70E C9          ret
```

```
******************************************* BASIC-Befehl RETURN
C70F  C0        ret   nz
C710  CD2EC7    call  C72E        GOSUB auf BASIC-Stack suchen
C713  CDACF5    call  F5AC        BASIC-Stackpointer zurücksetzen
C716  4E        ld    c,(hl)      Kennbyte
C717  23        inc   hl
C718  5E        ld    e,(hl)
C719  23        inc   hl          Adresse der Anweisung nach 'GOSUB'
C71A  56        ld    d,(hl)      nach de holen
C71B  23        inc   hl
C71C  7E        ld    a,(hl)
C71D  23        inc   hl          Zeilenadresse nach hl
C71E  66        ld    h,(hl)
C71F  6F        ld    l,a
C720  CDCEDD    call  DDCE        aktuelle Zeilennummer setzen
C723  EB        ex    de,hl       aktueller Programmzeiger nach hl
C724  79        ld    a,c         Kennbyte
C725  FE01      cp    01          kleiner eins ?
C727  D8        ret   c           ja, normales GOSUB
C728  CAA4C8    jp    z,C8A4      eins, dann GOSUB nach AFTER/EVERY
C72B  C3B6C8    jp    C8B6

*******************************************************
C72E  2A8BB0    ld    hl,(B08B)   BASIC-Stackpointer
C731  2B        dec   hl
C732  7E        ld    a,(hl)      Kennzeichen vom BASIC-Stack holen
C733  F5        push  af
C734  7D        ld    a,l
C735  96        sub   (hl)
C736  6F        ld    l,a
C737  9F        sbc   a,a
C738  84        add   a,h         BASIC-Stackpointer zurücksetzen
C739  67        ld    h,a
C73A  23        inc   hl
C73B  F1        pop   af
C73C  FE06      cp    06          'GOSUB'
C73E  C8        ret   z
C73F  B7        or    a
C740  20EF      jr    nz,C731
C742  1E03      ld    e,03        'Unexpected RETURN'
C744  C394CA    jp    CA94        Fehlermeldung ausgeben

******************************************* BASIC-Befehl WHILE
C747  E5        push  hl
C748  CD18CA    call  CA18        zugehöriges WEND suchen
C74B  E5        push  hl          Adresse merken
C74C  EB        ex    de,hl
C74D  222EAC    ld    (AC2E),hl   Zeilenadresse für WHILE-WEND
C750  CDB8C7    call  C7B8
C753  CCACF5    call  z,F5AC      BASIC-Stackpointer setzen
C756  3E07      ld    a,07        7 Bytes
C758  CDB0F5    call  F5B0        Platz im BASIC-Stack reservieren
C75B  EB        ex    de,hl
C75C  CDD2DD    call  DDD2        aktuelle Zeilenadresse nach hl
C75F  EB        ex    de,hl
```

```
C760  73          ld    (hl),e
C761  23          inc   hl          Zeilenadresse auf BASIC-Stack
C762  72          ld    (hl),d
C763  23          inc   hl
C764  D1          pop   de
C765  73          ld    (hl),e
C766  23          inc   hl          Adresse nach 'WEND' auf BASIC-Stack
C767  72          ld    (hl),d
C768  23          inc   hl
C769  EB          ex    de,hl
C76A  E3          ex    (sp),hl
C76B  EB          ex    de,hl
C76C  73          ld    (hl),e
C76D  23          inc   hl          Adresse der 'WHILE'-Bedingung
C76E  72          ld    (hl),d      auf BASIC-Stack
C76F  23          inc   hl
C770  3607        ld    (hl),07     Kennzeichen für 'WHILE'
C772  EB          ex    de,hl
C773  D1          pop   de
C774  182A        jr    C7A0        'WHILE'-Bedingung testen

***************************************** BASIC-Befehl WEND
C776  C0          ret   nz
C777  EB          ex    de,hl
C778  CDB8C7      call  C7B8
C77B  1E1E        ld    e,1E        'Unexpected WEND'
C77D  C294CA      jp    nz,CA94     Fehlermeldung ausgeben

C780  E5          push  hl
C781  110700      ld    de,0007
C784  19          add   hl,de
C785  CDACF5      call  F5AC        hl als BASIC-Stackpointer
C788  CDD2DD      call  DDD2        aktuelle Zeilenadresse nach hl
C78B  222EAC      ld    (AC2E),hl   Zeilenadresse für WHILE-WEND
C78E  E1          pop   hl
C78F  5E          ld    e,(hl)
C790  23          inc   hl
C791  56          ld    d,(hl)
C792  23          inc   hl
C793  EB          ex    de,hl
C794  CDCEDD      call  DDCE        aktuelle Zeilenadresse setzen
C797  EB          ex    de,hl
C798  5E          ld    e,(hl)
C799  23          inc   hl
C79A  56          ld    d,(hl)
C79B  23          inc   hl
C79C  7E          ld    a,(hl)
C79D  23          inc   hl
C79E  66          ld    h,(hl)
C79F  6F          ld    l,a
C7A0  D5          push  de
C7A1  CDFBCE      call  CEFB        Ausdruck holen
C7A4  E5          push  hl
C7A5  CDA3FD      call  FDA3        Vorzeichen holen
C7A8  E1          pop   hl
```

```
C7A9  D1              pop   de
C7AA  C0              ret   nz              Bedingung erfüllt ?
C7AB  2A2EAC          ld    hl,(AC2E)       Zeilenadresse für WHILE-WEND
C7AE  CDCEDD          call  DDCE            als aktuelle Zeilenadresse setzen
C7B1  3E07            ld    a,07
C7B3  CDA0F5          call  F5A0            Platz im BASIC-Stack freigeben
C7B6  EB              ex    de,hl
C7B7  C9              ret

*********************************************************
C7B8  2A8BB0          ld    hl,(B08B)       BASIC-Stackpointer
C7BB  2B              dec   hl
C7BC  E5              push  hl
C7BD  7D              ld    a,l
C7BE  96              sub   (hl)
C7BF  6F              ld    l,a
C7C0  9F              sbc   a,a
C7C1  84              add   a,h
C7C2  67              ld    h,a
C7C3  23              inc   hl
C7C4  E3              ex    (sp),hl
C7C5  7E              ld    a,(hl)
C7C6  FE10            cp    10              Integer 'FOR-NEXT'
C7C8  2816            jr    z,C7E0
C7CA  FE16            cp    16              Real 'FOR-NEXT'
C7CC  2812            jr    z,C7E0
C7CE  FE07            cp    07              'WHILE-WEND'
C7D0  200C            jr    nz,C7DE
C7D2  2B              dec   hl
C7D3  2B              dec   hl
C7D4  2B              dec   hl
C7D5  7E              ld    a,(hl)
C7D6  2B              dec   hl
C7D7  6E              ld    l,(hl)
C7D8  67              ld    h,a
C7D9  CDB8FF          call  FFB8            Vergleich hl <> de
C7DC  2002            jr    nz,C7E0
C7DE  E1              pop   hl
C7DF  C9              ret

C7E0  E1              pop   hl
C7E1  18D8            jr    C7BB

****************************************************** BASIC-Befehl ON
C7E3  FE9C            cp    9C              'ERROR'
C7E5  CAE5CB          jp    z,CBE5
C7E8  CD67CE          call  CE67            8-Bit-Wert holen
C7EB  4F              ld    c,a             als Zähler nach c
C7EC  46              ld    b,(hl)          Token holen
C7ED  78              ld    a,b
C7EE  FEA0            cp    A0              'GOTO'
C7F0  2805            jr    z,C7F7
C7F2  CD37DD          call  DD37            Test auf nachfolgendes Zeichen
C7F5  9F              db    9F              'GOSUB'
C7F6  2B              dec   hl
```

```
C7F7  0D        dec   c           Zähler erniedrigen
C7F8  78        ld    a,b         Token nach a
C7F9  CAABDD    jp    z,DDAB      BASIC-Befehl ausführen
C7FC  CD3FDD    call  DD3F        Blanks überlesen
C7FF  CDE1CE    call  CEE1        Zeilennummer nach de holen
C802  FE2C      cp    2C          ','
C804  28F1      jr    z,C7F7      nächste Zeilennummer
C806  C9        ret

************************************** Event-Verarbeitung (AFTER/EVERY)
C807  AF        xor   a
C808  3230AC    ld    (AC30),a
C80B  CDFBBC    call  BCFB        KL NEXT SYNC
C80E  301D      jr    nc,C82D     steht kein Ereignis an ?
C810  47        ld    b,a         Priorität merken
C811  3A30AC    ld    a,(AC30)
C814  E67F      and   7F          Bit 7 löschen
C816  3230AC    ld    (AC30),a
C819  C5        push  bc
C81A  E5        push  hl          Adresse des Event-Blocks
C81B  CDFEBC    call  BCFE        KL DO SYNC
C81E  E1        pop   hl
C81F  C1        pop   bc
C820  3A30AC    ld    a,(AC30)
C823  17        rla
C824  F5        push  af
C825  78        ld    a,b
C826  D401BD    call  nc,BD01     KL DONE SYNC
C829  F1        pop   af
C82A  17        rla
C82B  30DE      jr    nc,C80B     nächstes Event
C82D  3A30AC    ld    a,(AC30)
C830  E604      and   04
C832  C453C4    call  nz,C453     Unterbrechung durch 'ESC' erlauben
C835  2A34AE    ld    hl,(AE34)   Adresse des aktuellen Statements
C838  3A30AC    ld    a,(AC30)
C83B  E603      and   03
C83D  C8        ret   z
C83E  1F        rra
C83F  DA6BCB    jp    c,CB6B      'Break'
C842  23        inc   hl
C843  F1        pop   af
C844  C393DD    jp    DD93        zur Interpreterschleife

****************************************************************
C847  2236AC    ld    (AC36),hl
C84A  3E04      ld    a,04
C84C  3050      jr    nc,C89E
C84E  2A34AC    ld    hl,(AC34)   ON-BREAK-Adresse
C851  7C        ld    a,h
C852  B5        or    l
C853  C4D6DD    call  nz,DDD6     Zeilennummer nach hl
C856  3E41      ld    a,41
C858  3044      jr    nc,C89E     Direktmodus ?
C85A  1131AC    ld    de,AC31
```

```
C85D 0E02       ld     c,02
C85F 1825       jr     C886

*************************************************************
C861 D5         push   de
C862 CD37DD     call   DD37         Test auf nachfolgendes Zeichen
C865 9F         db     9F           'GOSUB'
C866 CD67E7     call   E767         Zeilenadresse holen
C869 42         ld     b,d
C86A 4B         ld     c,e
C86B CD61DD     call   DD61         Blank, TAB und LF überlesen
C86E D1         pop    de
C86F E5         push   hl
C870 210A00     ld     hl,000A      10
C873 19         add    hl,de
C874 71         ld     (hl),c
C875 23         inc    hl
C876 70         ld     (hl),b
C877 E1         pop    hl
C878 C9         ret

****************************************************** Event-Routine
C879 23         inc    hl
C87A 23         inc    hl
C87B 23         inc    hl
C87C EB         ex     de,hl
C87D CDD6DD     call   DDD6         Zeilennummer holen/Direktmodus ?
C880 3E40       ld     a,40
C882 301A       jr     nc,C89E      ja
C884 0E01       ld     c,01         Kennbyte für AFTER/EVERY GOSUB
C886 D5         push   de
C887 CDF6C6     call   C6F6         GOSUB-Befehl
C88A 2A34AE     ld     hl,(AE34)    Adresse des aktuellen Statements
C88D EB         ex     de,hl
C88E E1         pop    hl
C88F 70         ld     (hl),b
C890 23         inc    hl
C891 73         ld     (hl),e
C892 23         inc    hl
C893 72         ld     (hl),d
C894 23         inc    hl
C895 5E         ld     e,(hl)
C896 23         inc    hl
C897 56         ld     d,(hl)
C898 EB         ex     de,hl
C899 2234AE     ld     (AE34),hl    Adresse des aktuellen Statements
C89C 3EC2       ld     a,C2
C89E 2130AC     ld     hl,AC30
C8A1 B6         or     (hl)
C8A2 77         ld     (hl),a
C8A3 C9         ret
```

```
C8A4 7E          ld    a,(hl)
C8A5 23          inc   hl
C8A6 5E          ld    e,(hl)
C8A7 23          inc   hl
C8A8 56          ld    d,(hl)
C8A9 D5          push  de
C8AA 01F7FF      ld    bc,FFF7
C8AD 09          add   hl,bc
C8AE CD01BD      call  BD01        KL DONE SYNC
C8B1 E1          pop   hl
C8B2 F1          pop   af
C8B3 C374DD      jp    DD74        zur Interpreterschleife

C8B6 7E          ld    a,(hl)
C8B7 2A36AC      ld    hl,(AC36)
C8BA 01FCFF      ld    bc,FFFC
C8BD 09          add   hl,bc
C8BE CD01BD      call  BD01        KL DONE SYNC
C8C1 CD53C4      call  C453        Unterbrechung durch 'Break' erlauben
C8C4 2A32AC      ld    hl,(AC32)
C8C7 F1          pop   af
C8C8 C374DD      jp    DD74        zur Interpreterschleife

******************************************** BASIC-Befehl ON BREAK
C8CB FECE        cp    CE          'STOP'
C8CD 110000      ld    de,0000     Defaultwert 0 bei STOP
C8D0 2808        jr    z,C8DA
C8D2 CD37DD      call  DD37        Test auf nachfolgendes Zeichen
C8D5 9F          db    9F          'GOSUB'
C8D6 CD67E7      call  E767        Zeilenadresse holen
C8D9 2B          dec   hl
C8DA ED5334AC    ld    (AC34),de   ON-BREAK Adresse
C8DE C33FDD      jp    DD3F        Blanks überlesen

************************************************** BASIC-Befehl DI
C8E1 E5          push  hl
C8E2 CD04BD      call  BD04        KL EVENT DISABLE
C8E5 E1          pop   hl
C8E6 C9          ret

************************************************** BASIC-Befehl EI
C8E7 E5          push  hl
C8E8 CD07BD      call  BD07        KL EVENT ENABLE
C8EB E1          pop   hl
C8EC C9          ret

******************************************** SOUND und Event-Reset
C8ED CDA7BC      call  BCA7        SOUND RESET
C8F0 215CAC      ld    hl,AC5C     Basis-Adresse der Event-Blocks
C8F3 0604        ld    b,04        4 Timer
C8F5 E5          push  hl
C8F6 CDECBC      call  BCEC        KL DEL TICKER
C8F9 E1          pop   hl
C8FA 111200      ld    de,0012     18
C8FD 19          add   hl,de       addieren
```

```
C8FE 10F5      djnz  C8F5          nächster Timer
C900 CD48BB    call  BB48          KM DISARM BREAK
C903 CDF5BC    call  BCF5          KL SYNC RESET
C906 210000    ld    hl,0000
C909 2234AC    ld    (AC34),hl     ON-BREAK Adresse löschen
C90C CD53C4    call  C453          Unterbrechung durch 'Break' erlauben
C90F 2138AC    ld    hl,AC38
C912 110503    ld    de,0305
C915 010008    ld    bc,0800
C918 CD24C9    call  C924
C91B 2162AC    ld    hl,AC62       Adresse des Event-Blocks
C91E 110B04    ld    de,040B
C921 010102    ld    bc,0201
C924 C5        push  bc
C925 D5        push  de
C926 0EFD      ld    c,FD          BASIC-ROM Selekt
C928 1179C8    ld    de,C879       Adresse der Event-Routine
C92B CDEFBC    call  BCEF          KL INIT EVENT
C92E D1        pop   de
C92F D5        push  de
C930 1600      ld    d,00
C932 19        add   hl,de
C933 D1        pop   de
C934 C1        pop   bc
C935 79        ld    a,c
C936 B7        or    a
C937 2803      jr    z,C93C
C939 78        ld    a,b
C93A 87        add   a,a
C93B 47        ld    b,a
C93C 15        dec   d
C93D 20E5      jr    nz,C924
C93F C9        ret

**************************************** BASIC-Befehl ON SQ
C940 CD37DD    call  DD37          Test auf nachfolgendes Zeichen
C943 28        db    28            '('
C945 CD67CE    call  CE67          8-Bit-Wert holen
C947 F5        push  hl
C948 CD5DC9    call  C95D          Adresse der Sound-Queue berechnen
C94B B7        or    a             größer 4 ?
C94C 201E      jr    nz,C96C       'Improper argument'
C94E CD37DD    call  DD37          Test auf nachfolgendes Zeichen
C951 29        db    29            ')'
C952 CD61C8    call  C861          'GOSUB' und Adresse holen
C955 F1        pop   af
C956 E5        push  hl
C957 EB        ex    de,hl
C958 CDB0BC    call  BCB0          SOUND ARM EVENT
C95B E1        pop   hl
C95C C9        ret
```

```
************************************** Adresse der Sound-Queue berechnen
C95D 1F          rra                  Bit 0 gesetzt ?
C95E 1138AC      ld     de,AC38
C961 D8          ret    c
C962 1F          rra                  Bit 1 gesetzt ?
C963 1144AC      ld     de,AC44
C966 D8          ret    c
C967 1F          rra                  Bit 2 gesetzt ?
C968 1150AC      ld     de,AC50
C96B D8          ret    c
C96C 1E05        ld     e,05          'Improper argument'
C96E C394CA      jp     CA94          Fehlermeldung ausgeben

********************************************** BASIC-Befehl AFTER
C971 CD7CCE      call   CE7C          16-Bit-Wert 0 - 32767 holen
C974 010000      ld     bc,0000       Recharge Count auf Null
C977 1805        jr     C97E

********************************************** BASIC-Befehl EVERY
C979 CD7CCE      call   CE7C          16-Bit-Wert 0 - 32767 holen
C97C 42          ld     b,d           als Count und
C97D 4B          ld     c,e           Recharge Count
C97E D5          push   de
C97F C5          push   bc
C980 CD55DD      call   DD55          folgt Komma ?
C983 110000      ld     de,0000       Defaultwert Null
C986 DC86CE      call   c,CE86        ja, Integerwert mit Vorzeichen holen
C989 EB          ex     de,hl
C98A CDB1C9      call   C9B1          Aus Timer# Adresse des Event-Blocks holen
C98D E5          push   hl
C98E 010600      ld     bc,0006       6 Bytes für Tickerblock addieren
C991 09          add    hl,bc
C992 EB          ex     de,hl
C993 CD61C8      call   C861          'GOSUB' und Adresse holen
C996 D1          pop    de
C997 C1          pop    bc
C998 E3          ex     (sp),hl
C999 EB          ex     de,hl
C99A CDE9BC      call   BCE9          KL ADD TICKER
C99D E1          pop    hl
C99E C9          ret

******************************************** BASIC-Funktion REMAIN
C99F CD8DFE      call   FE8D          CINT
C9A2 CDB1C9      call   C9B1          Adresse des Event-Blocks holen
C9A5 CDECBC      call   BCEC          KL DEL TICKER
C9A8 3803        jr     c,C9AD        gefunden ?
C9AA 110000      ld     de,0000       nein, Null
C9AD EB          ex     de,hl
C9AE C30DFF      jp     FF0D          Integerzahl in hl Übernehmen

************************************** Adresse des Event-Blocks berechnen
C9B1 7C          ld     a,h
C9B2 B7          or     a             Hi-Byte ungleich Null ?
C9B3 20B7        jr     nz,C96C       ja, 'Improper argument'
```

```
C9B5 7D          ld    a,l
C9B6 FE04        cp    04          größer gleich 4 ?
C9B8 30B2        jr    nc,C96C     ja, 'Improper argument'
C9BA 87          add   a,a
C9BB 87          add   a,a
C9BC 87          add   a,a         * 18
C9BD 85          add   a,l
C9BE 87          add   a,a
C9BF 6F          ld    l,a
C9C0 015CAC      ld    bc,AC5C     Basisadresse Event-Tabelle
C9C3 09          add   hl,bc       plus Offset
C9C4 C9          ret

***************************************** zugehöriges NEXT suchen
C9C5 EB          ex    de,hl
C9C6 CDD2DD      call  DDD2        aktuelle Zeilenadresse nach hl
C9C9 EB          ex    de,hl
C9CA 2B          dec   hl
C9CB 0601        ld    b,01        Zähler für Verschachtelung
C9CD 0E1A        ld    c,1A        Fehlernummer für 'NEXT missing'
C9CF CD23E9      call  E923
C9D2 E5          push  hl
C9D3 CD3FDD      call  DD3F        Blanks überlesen
C9D6 FEB0        cp    B0          'NEXT'
C9D8 2808        jr    z,C9E2
C9DA E1          pop   hl
C9DB FE9E        cp    9E          'FOR'
C9DD 20EE        jr    nz,C9CD
C9DF 04          inc   b           Verschachtelung erhöhen
C9E0 18EB        jr    C9CD        weiter suchen

C9E2 F1          pop   af
C9E3 EB          ex    de,hl
C9E4 E5          push  hl
C9E5 CDD2DD      call  DDD2        aktuelle Zeilenadresse nach hl
C9E8 E3          ex    (sp),hl
C9E9 CDCEDD      call  DDCE        aktuelle Zeilenadresse setzen
C9EC EB          ex    de,hl
C9ED 05          dec   b           Verschachtelung erniedrigen
C9EE 2824        jr    z,CA14      zugehöriges 'NEXT' gefunden ?
C9F0 CD3FDD      call  DD3F        Blanks überlesen
C9F3 280E        jr    z,CA03      Zeilenende ?
C9F5 C5          push  bc
C9F6 D5          push  de
C9F7 CD86D6      call  D686        Variable suchen
C9FA D1          pop   de
C9FB C1          pop   bc
C9FC CD55DD      call  DD55        folgt Komma ?
C9FF 3002        jr    nc,CA03     nein
CA01 10F2        djnz  C9F5        sonst nächste Variable nach 'NEXT'
CA03 2B          dec   hl
CA04 78          ld    a,b         zugehöriges 'NEXT' gefunden ?
CA05 B7          or    a
CA06 280C        jr    z,CA14      ja
CA08 EB          ex    de,hl
```

```
CA09  CDD2DD      call   DDD2      aktuelle Zeilenadresse nach hl
CA0C  E3          ex     (sp),hl
CA0D  CDCEDD      call   DDCE      aktuelle Zeilenadresse setzen
CA10  E1          pop    hl
CA11  EB          ex     de,hl
CA12  18B9        jr     C9CD      weitersuchen

CA14  D1          pop    de
CA15  C33FDD      jp     DD3F      Blanks überlesen

****************************************** zugehöriges WEND suchen
CA18  2B          dec    hl
CA19  EB          ex     de,hl
CA1A  CDD2DD      call   DDD2      aktuelle Zeilenadresse nach hl
CA1D  EB          ex     de,hl
CA1E  0600        ld     b,00      Zähler für Verschachtelung
CA20  04          inc    b
CA21  0E1D        ld     c,1D      Fehlernummer für 'WEND missing'
CA23  CD23E9      call   E923
CA26  E5          push   hl
CA27  CD3FDD      call   DD3F      Blanks überlesen
CA2A  E1          pop    hl
CA2B  FED6        cp     D6        'WHILE'
CA2D  28F1        jr     z,CA20    Verschachtelung erhöhen
CA2F  FED5        cp     D5        'WEND'
CA31  20EE        jr     nz,CA21
CA33  10EC        djnz   CA21      Verschachtelung erniedrigen
CA35  CD3FDD      call   DD3F      Blanks überlesen
CA38  C33FDD      jp     DD3F      Blanks überlesen

********************************************* Eingabezeile holen
CA3B  21A4AC      ld     hl,ACA4   Zeiger auf Eingabepuffer
CA3E  3600        ld     (hl),00   Pufferinhalt löschen
CA40  C33ABD      jp     BD3A      Eingabezeile holen

************************************************* Zeile editieren
CA43  21A4AC      ld     hl,ACA4   Zeiger auf Eingabepuffer
CA46  CD3ABD      call   BD3A      Zeile editieren
CA49  C34EC3      jp     C34E      LF ausgeben

************************************** Eingabezeile von Kassette holen
CA4C  C5          push   bc
CA4D  D5          push   de
CA4E  21A4AC      ld     hl,ACA4   Zeiger auf Eingabepuffer
CA51  E5          push   hl
CA52  0601        ld     b,01
CA54  0E00        ld     c,00
CA56  CD80BC      call   BC80      CAS IN CHAR
CA59  CA6BCB      jp     z,CB6B
CA5C  3022        jr     nc,CA80
CA5E  77          ld     (hl),a
CA5F  FE0D        cp     0D        CR
CA61  2817        jr     z,CA7A
CA63  0E00        ld     c,00
CA65  FE0A        cp     0A        LF
```

```
CA67  2006          jr    nz,CA6F
CA69  78            ld    a,b
CA6A  3D            dec   a
CA6B  28E7          jr    z,CA54
CA6D  0EFF          ld    c,FF
CA6F  78            ld    a,b
CA70  B7            or    a
CA71  1E17          ld    e,17          'Line too long'
CA73  CA94CA        jp    z,CA94        Fehlermeldung ausgeben

CA76  23            inc   hl
CA77  04            inc   b
CA78  18DC          jr    CA56

CA7A  79            ld    a,c
CA7B  B7            or    a
CA7C  20D8          jr    nz,CA56
CA7E  77            ld    (hl),a
CA7F  37            scf
CA80  E1            pop   hl
CA81  D1            pop   de
CA82  C1            pop   bc
CA83  C9            ret

*************************************** Fehlernummer löschen
CA84  AF            xor   a

*************************************** Fehlernummer setzen
CA85  32AAAD        ld    (ADAA),a      Fehlernummer
CA88  CDD2DD        call  DDD2          aktuelle Zeilenadresse nach hl
CA8B  22A6AD        ld    (ADA6),hl     ERROR-Line
CA8E  C9            ret

*************************************** BASIC-Befehl ERROR
CA8F  CD6DCE        call  CE6D          8-Bit-Wert ungleich Null holen
CA92  C0            ret   nz
CA93  5F            ld    e,a           Fehlernummer nach e

*************************************** Fehlermeldung ausgeben
CA94  CD04AC        call  AC04          ret
CA97  7B            ld    a,e
CA98  CD85CA        call  CA85          Fehlernummer und -zeile merken
CA9B  2A34AE        ld    hl,(AE34)     Adresse des aktuellen Statements
CA9E  22A8AD        ld    (ADA8),hl     Programmzeiger nach ERROR
CAA1  CDB0CB        call  CBB0          Zeilenadresse und Programmzeiger merken
CAA4  3100C0        ld    sp,C000       Stackpointer auf C000
CAA7  2A32AE        ld    hl,(AE32)     Speicher für BASIC-Stackpointer
CAAA  CDACF5        call  F5AC          BASIC-Stackpointer neu setzen
CAAD  CDB3FB        call  FBB3          Descriptorstack initialisieren
CAB0  CDFDD9        call  D9FD          AE29 und AE2B löschen
CAB3  CDDFCA        call  CADF          Zeilennummer der ERROR-Zeile holen
CAB6  2AAFAD        ld    hl,(ADAF)     Adresse der ON-ERROR-Routine
CAB9  EB            ex    de,hl
CABA  21B1AD        ld    hl,ADB1       Flag für in Fehlerbehandlung
CABD  300C          jr    nc,CACB
```

```
CABF 7A        ld    a,d
CAC0 B3        or    e
CAC1 2808      jr    z,CACB
CAC3 A6        and   (hl)
CAC4 2005      jr    nz,CACB
CAC6 35        dec   (hl)
CAC7 EB        ex    de,hl
CAC8 C393DD    jp    DD93          zur Interpreterschleife

CACB 3600      ld    (hl),00
CACD 3AAAAD    ld    a,(ADAA)      ERROR-Nummer
CAD0 CD45CC    call  CC45          Zeiger auf Fehlermeldung setzen
CAD3 2AA6AD    ld    hl,(ADA6)     Adresse der ERROR-Zeile
CAD6 CDCEDD    call  DDCE          aktuelle Zeilenadresse setzen
CAD9 CD36CB    call  CB36
CADC C364C0    jp    C064          zum READY-Modus

***********************************************************************
CADF 2AA6AD    ld    hl,(ADA6)     Adresse der ERROR-Zeile
CAE2 CDD9DD    call  DDD9          Zeilennummer nach hl holen
CAE5 D8        ret   c
CAE6 210000    ld    hl,0000
CAE9 C9        ret

***********************************************************************
CAEA D5        push  de
CAEB E5        push  hl
CAEC 2113CD    ld    hl,CD13       Zeiger auf
CAEF 1E0B      ld    e,0B          'Division by zero'
CAF1 1807      jr    CAFA

CAF3 D5        push  de
CAF4 E5        push  hl
CAF5 21B9CC    ld    hl,CCB9       Zeiger auf
CAF8 1E06      ld    e,06          'Overflow'
CAFA F5        push  af
CAFB E5        push  hl
CAFC 2AAFAD    ld    hl,(ADAF)     Adresse der ON-ERROR-Routine
CAFF 7C        ld    a,h
CB00 B5        or    l
CB01 E1        pop   hl
CB02 C294CA    jp    nz,CA94       Fehlermeldung ausgeben

CB05 AF        xor   a
CB06 CDA2C1    call  C1A2
CB09 F5        push  af
CB0A CD41C3    call  C341          String ausgeben
CB0D CD4EC3    call  C34E          LF ausgeben
CB10 F1        pop   af
CB11 CDA2C1    call  C1A2
CB14 F1        pop   af
CB15 E1        pop   hl
CB16 D1        pop   de
CB17 C9        ret
```

```
CB18  CD86C3         call   C386         Bildschirm initialisieren
CB1B  2123CB         ld     hl,CB23      'Undefined line' ausgeben
CB1E  CD48CB         call   CB48         Zeilennummer ausgeben
CB21  181D           jr     CB40         'in Zeilennummer' ausgeben

************************************************************
CB23  55 6E 64 65 66 69 6E 65
CB2B  64 20 6C 69 6E 65 20 00            'Undefined line '

************************************************************
CB33  114FCB         ld     de,CB4F      Zeiger auf 'Break'
CB36  CD9DC1         call   C19D
CB39  CD86C3         call   C386         Bildschirm initialisieren
CB3C  EB             ex     de,hl
CB3D  CD41C3         call   C341         ausgeben
CB40  CDD6DD         call   DDD6         Zeilennummer nach hl holen
CB43  D0             ret    nc           Direktmodus ?
CB44  EB             ex     de,hl
CB45  2155CB         ld     hl,CB55      Zeiger auf ' in '
CB48  CD41C3         call   C341         ausgeben
CB4B  EB             ex     de,hl
CB4C  C379EE         jp     EE79         Zeilennummer ausgeben

************************************************************
CB4F  42 72 65 61 6B 00                  'Break'
CB55  20 69 6E 20 00                     ' in '

******************************************************* BASIC-Befehl STOP
CB5A  C0             ret    nz
CB5B  E5             push   hl
CB5C  CD33CB         call   CB33         'Break in Zeilennummer'
CB5F  E1             pop    hl
CB60  CD93CB         call   CB93
CB63  182B           jr     CB90         zum READY-Modus

******************************************************* BASIC-Befehl END
CB65  C0             ret    nz
CB66  CD93CB         call   CB93
CB69  181C           jr     CB87

CB6B  CD33CB         call   CB33
CB6E  2A34AE         ld     hl,(AE34)    Adresse des aktuellen Statements
CB71  CDB0CB         call   CBB0
CB74  181A           jr     CB90

CB76  CDD6DD         call   DDD6         Direkt-Modus ?
CB79  3012           jr     nc,CB8D      ja
CB7B  CDABCB         call   CBAB
CB7E  3AB1AD         ld     a,(ADB1)     noch in Fehlerbehandlung ?
CB81  B7             or     a
CB82  1E13           ld     e,13         'RESUME missing'
CB84  C294CA         jp     nz,CA94      ja, Fehlermeldung ausgeben
```

```
CB87  CD98D2      call   D298
CB8A  CDA1D2      call   D2A1
CB8D  CDCBDD      call   DDCB        aktuelle Zeilenadresse auf Null
CB90  C364C0      jp     C064        zum READY-Modus

**********************************************************
CB93  EB          ex     de,hl
CB94  CDD6DD      call   DDD6        Zeilennummer nach hl holen
CB97  EB          ex     de,hl
CB98  D0          ret    nc          Direktmodus ?
CB99  7E          ld     a,(hl)
CB9A  FE01        cp     01
CB9C  280B        jr     z,CBA9
CB9E  23          inc    hl
CB9F  7E          ld     a,(hl)
CBA0  23          inc    hl
CBA1  B6          or     (hl)
CBA2  2807        jr     z,CBAB
CBA4  23          inc    hl
CBA5  CDCEDD      call   DDCE        aktuelle Zeilenadresse setzen
CBA8  23          inc    hl
CBA9  1805        jr     CBB0

CBAB  210000      ld     hl,0000
CBAE  180C        jr     CBBC

CBB0  EB          ex     de,hl
CBB1  CDD6DD      call   DDD6        Direktmodus ?
CBB4  D0          ret    nc          ja
CBB5  CDD2DD      call   DDD2        Zeilenadresse nach hl
CBB8  22ADAD      ld     (ADAD),hl   Zeilenadresse nach Unterbrechung
CBBB  EB          ex     de,hl
CBBC  22ABAD      ld     (ADAB),hl   Programmzeiger nach Unterbrechung
CBBF  C9          ret

**************************************************** BASIC-Befehl CONT
CBC0  C0          ret    nz
CBC1  2AABAD      ld     hl,(ADAB)   Programmzeiger nach Unterbrechung
CBC4  7C          ld     a,h
CBC5  B5          or     l
CBC6  1E11        ld     e,11        'Cannot CONTinue'
CBC8  CA94CA      jp     z,CA94      Fehlermeldung ausgeben

CBCB  E5          push   hl
CBCC  2AADAD      ld     hl,(ADAD)   Zeilenadresse nach Unterbrechung
CBCF  CDCEDD      call   DDCE        aktuelle Zeilenadresse setzen
CBD2  CDB9BC      call   BCB9        SOUND CONTINUE
CBD5  E1          pop    hl
CBD6  C374DD      jp     DD74        zur Interpreterschleife

**********************************************************
CBD9  AF          xor    a
CBDA  32B1AD      ld     (ADB1),a    Flag für in Fehlerbehandlung löschen
CBDD  110000      ld     de,0000
CBE0  ED53AFAD    ld     (ADAF),de   Adresse der ON-ERROR-Routine
```

```
CBE4  C9             ret

********************************************************* ON ERROR
CBE5  CD3FDD         call  DD3F          Blanks überlesen
CBE8  CD37DD         call  DD37          Test auf nachfolgendes Zeichen
CBEB  A0             db    A0            'GOTO'
CBEC  CDE1CE         call  CEE1          Zeilennummer nach de holen
CBEF  E5             push  hl
CBF0  CD9AE7         call  E79A          BASIC-Zeile de suchen
CBF3  22AFAD         ld    (ADAF),hl     Adresse der ON-ERROR-Routine
CBF6  E1             pop   hl
CBF7  C9             ret

**************************************** BASIC-Befehl ON ERROR GOTO 0
CBF8  CDDDCB         call  CBDD          aktuelle Zeilenadresse auf Null
CBFB  3AB1AD         ld    a,(ADB1)      in Fehlerbehandlung ?
CBFE  B7             or    a
CBFF  C8             ret   z             nein
CC00  C3A4CA         jp    CAA4

********************************************** BASIC-Befehl RESUME
CC03  2814           jr    z,CC19
CC05  FEB0           cp    B0            'NEXT'
CC07  2817           jr    z,CC20
CC09  CD67E7         call  E767          Zeilenadresse holen
CC0C  CD4ADD         call  DD4A          Ende des Statements, sonst 'Syntax error'
CC0F  D5             push  de
CC10  CD2BCC         call  CC2B          ERROR-Flags löschen
CC13  E1             pop   hl
CC14  23             inc   hl
CC15  F1             pop   af
CC16  C393DD         jp    DD93          zur Interpreterschleife

CC19  CD2BCC         call  CC2B          ERROR-Flags löschen
CC1C  F1             pop   af
CC1D  C374DD         jp    DD74          zur Interpreterschleife

CC20  CD3FDD         call  DD3F          Blanks überlesen
CC23  C0             ret   nz
CC24  CD2BCC         call  CC2B          ERROR-Flags löschen
CC27  23             inc   hl
CC28  C3EFE8         jp    E8EF          Rest der Zeile überlesen

CC2B  3AB1AD         ld    a,(ADB1)      in Fehlerbehandlung ?
CC2E  B7             or    a
CC2F  1E14           ld    e,14          'Unexpected RESUME'
CC31  CA94CA         jp    z,CA94        nein, Fehlermeldung ausgeben

CC34  AF             xor   a
CC35  32AAAD         ld    (ADAA),a      ERROR-Nummer löschen
CC38  32B1AD         ld    (ADB1),a      Flag für in Fehlerbehandlung löschen
CC3B  2AA6AD         ld    hl,(ADA6)     Adresse der ERROR-Zeile
CC3E  CDCEDD         call  DDCE          als aktuelle Zeilenadresse
CC41  2AA8AD         ld    hl,(ADA8)     Programmzeiger nach ERROR
CC44  C9             ret
```

```
****************************************** Zeiger auf Fehlermeldung setzen
CC45  115BCC      ld    de,CC5B      Basisadresse der Fehlermeldungen
CC48  FE1F        cp    1F           31, max. Fehlernummer +1
CC4A  D0          ret   nc           >=, dann 0 'Unknown error'
CC4B  B7          or    a
CC4C  C8          ret   z            0, dann fertig
CC4D  47          ld    b,a          Fehlernummer nach b
CC4E  1A          ld    a,(de)
CC4F  13          inc   de           Zeichen lesen
CC50  B7          or    a            bis 0, Ende einer Meldung
CC51  20FB        jr    nz,CC4E
CC53  05          dec   b            nächste Meldung
CC54  20F8        jr    nz,CC4E      noch nicht richtige Meldung ?
CC56  1A          ld    a,(de)
CC57  B7          or    a
CC58  28EB        jr    z,CC45
CC5A  C9          ret

***************************************************** Fehlermeldungen
CC5B 55 6E 6B 6E 6F 77 6E 20 65
CC64 72 72 6F 72 00                  0    Unknown error
CC69 55 6E 65 78 70 65 63 74 65
CC72 64 20 4E 45 58 54 00            1    Unexpected NEXT
CC79 53 79 6E 74 61 78 20 65 72
CC82 72 6F 72 00                     2    Syntax error
CC86 55 6E 65 78 70 65 63 74 65
CC8F 64 20 52 45 54 55 52 4E 00      3    Unexpected RETURN
CC98 44 41 54 41 20 65 78 68 61
CCA1 75 73 74 65 64 00               4    DATA exhausted
CCA7 49 6D 70 72 6F 70 65 72 20
CCB0 61 72 67 75 6D 65 6E 74 00      5    Improper argument
CCB9 4F 76 65 72 66 6C 6F 77 00      6    Overflow
CCC2 4D 65 6D 6F 72 79 20 66 75
CCCB 6C 6C 00                        7    Memory full
CCCE 4C 69 6E 65 20 64 6F 65 73
CCD7 20 6E 6F 74 20 65 78 69 73
CCE0 74 00                           8    Line does not exist
CCE2 53 75 62 73 63 72 69 70 74
CCEB 20 6F 75 74 20 6F 66 20 72
CCF4 61 6E 67 65 00                  9    Subscript out of range
CCF9 41 72 72 61 79 20 61 6C 72
CD02 65 61 64 79 20 64 69 6D 65
CD0B 6E 73 69 6F 6E 65 64 00         10   Array already dimensioned
CD13 44 69 76 69 73 69 6F 6E 20
CD1C 62 79 20 7A 65 72 6F 00         11   Division by zero
CD24 49 6E 76 61 6C 69 64 20 64
CD2D 69 72 65 63 74 20 63 6F 6D
CD35 6D 61 6E 64 00                  12   Invalid direct command
CD3B 54 79 70 65 20 6D 69 73 6D
CD44 61 74 63 68 00                  13   Type mismatch
CD49 53 74 72 69 6E 67 20 73 70
CD52 61 63 65 20 66 75 6C 6C 00      14   String space full
CD5B 53 74 72 69 6E 67 20 74 6F
CD64 6F 20 6C 6F 6E 67 00            15   String too long
CD6B 53 74 72 69 6E 67 20 65 78
```

```
CD74 70 72 65 73 73 69 6F 6E 20
CD7D 74 6F 6F 20 63 6F 6D 70 6C
CD86 65 78 00                            16  String expression too complex
CD89 43 61 6E 6E 6F 74 20 43 4F
CD92 4E 54 69 6E 75 65 00                17  Cannot CONTinue
CD99 55 6E 6B 6E 6F 77 6E 20 75
CDA2 73 65 72 20 66 75 6E 63 74
CDAB 69 6F 6E 00                         18  Unknown user function
CDAF 52 45 53 55 4D 45 20 6D 69
CDB8 73 73 69 6E 67 00                   19  RESUME missing
CDBE 55 6E 65 78 70 65 63 74 65
CDC7 64 20 52 45 53 55 4D 45 00          20  Unexpected RESUME
CDD0 44 69 72 65 63 74 20 63 6F
CDD9 6D 6D 61 6E 64 20 66 6F 75
CDE2 6E 64 00                            21  Direct command found
CDE5 4F 70 65 72 61 6E 64 20 6D
CDEE 69 73 73 69 6E 67 00                22  Operand missing
CDF5 4C 69 6E 65 20 74 6F 6F 20
CDFE 6C 6F 6E 67 00                      23  Line too long
CE03 45 4F 46 20 6D 65 74 00             24  EOF met
CE0B 46 69 6C 65 20 74 79 70 65
CE14 20 65 72 72 6F 72 00                25  File type error
CE1B 4E 45 58 54 20 6D 69 73 73
CE24 69 6E 67 00                         26  NEXT missing
CE28 46 69 6C 65 20 61 6C 72 65
CE31 61 64 79 20 6F 70 65 6E 00          27  File already open
CE3A 55 6E 6B 6E 6F 77 6E 20 63
CE43 6F 6D 6D 61 6E 64 00                28  Unknown command
CE4A 57 45 4E 44 20 6D 69 73 73
CE53 69 6E 67 00                         29  WEND missing
CE57 55 6E 65 78 70 65 63 74 65
CE60 64 20 57 45 4E 44 00                30  Unexpected WEND

****************************************************** 8-Bit-Wert holen
CE67 CD86CE        call  CE86        Integerwert mit Vorzeichen holen
CE6A F5            push  af
CE6B 1808          jr    CE75

******************************************** 8-Bit-Wert ungleich Null holen
CE6D CD86CE        call  CE86        Integerwert mit Vorzeichen holen
CE70 F5            push  af
CE71 7A            ld    a,d
CE72 B3            or    e           Null ?
CE73 2836          jr    z,CEAB      dann 'Improper argument'
CE75 7A            ld    a,d
CE76 B7            or    a           Hi-Byte ungleich Null ?
CE77 2032          jr    nz,CEAB     dann 'Improper argument'
CE79 F1            pop   af
CE7A 7B            ld    a,e
CE7B C9            ret

******************************************** 16-Bit-Wert 0 bis 32767 holen
CE7C CD86CE        call  CE86        Integerwert mit Vorzeichen holen
CE7F F5            push  af
CE80 7A            ld    a,d
```

```
CE81  17            rla                      Bit 15 gesetzt ?
CE82  3827          jr      c,CEAB           dann 'Improper argument'
CE84  F1            pop     af
CE85  C9            ret

*************************************** Integerwert mit Vorzeichen holen
CE86  CDFBCE        call    CEFB             Ausdruck holen
CE89  F5            push    af
CE8A  EB            ex      de,hl
CE8B  CD8DFE        call    FE8D             CINT
CE8E  EB            ex      de,hl
CE8F  F1            pop     af
CE90  C9            ret

*************************************** 16-Bit-Wert holen, Adreßausdruck
CE91  CDFBCE        call    CEFB             Ausdruck holen
CE94  F5            push    af
CE95  C5            push    bc
CE96  E5            push    hl
CE97  CDC2FE        call    FEC2             UNT
CE9A  EB            ex      de,hl
CE9B  E1            pop     hl
CE9C  C1            pop     bc
CE9D  F1            pop     af
CE9E  C9            ret

************************************** Stringausdruck und Parameter holen
CE9F  CDFBCE        call    CEFB             Ausdruck holen
CEA2  C3DAFB        jp      FBDA             Stringparameter holen

********************************************** Stringausdruck holen
CEA5  CDFBCE        call    CEFB             Ausdruck holen
CEA8  C33CFF        jp      FF3C             Typ 'String', sonst 'Type mismatch'

CEAB  1E05          ld      e,05             'Improper argument'
CEAD  C394CA        jp      CA94             Fehlermeldung ausgeben

****************************************** Zeilennummernbereich holen
CEB0  010100        ld      bc,0001          1
CEB3  11FFFF        ld      de,FFFF          und 65535 als Default
CEB6  CD55DD        call    DD55             auf Komma prüfen
CEB9  D451DD        call    nc,DD51          Ende des Statements ?
CEBC  D8            ret     c                ja
CEBD  FE23          cp      23               '#' (Kanalnummer) ?
CEBF  C8            ret     z
CEC0  FEF5          cp      F5               '-'
CEC2  280A          jr      z,CECE
CEC4  CDE1CE        call    CEE1             Zeilennummer nach de holen
CEC7  42            ld      b,d
CEC8  4B            ld      c,e              nach bc kopieren
CEC9  C8            ret     z
CECA  CD55DD        call    DD55             folgt Komma ?
CECD  D8            ret     c                ja
CECE  CD37DD        call    DD37             Test auf nachfolgendes Zeichen
CED1  F5            db      F5               '-'
```

```
CED2 11FFFF      ld    de,FFFF     65535 als Default-Endwert
CED5 C8          ret   z
CED6 CD55DD      call  DD55        auf Komma prüfen
CED9 D8          ret   c
CEDA CDE1CE      call  CEE1        Zeilennummer nach de holen
CEDD C455DD      call  nz,DD55     auf Komma prüfen
CEE0 C9          ret

*************************************** Zeilennummmer nach de holen
CEE1 7E          ld    a,(hl)      Konstantentyp
CEE2 23          inc   hl
CEE3 5E          ld    e,(hl)
CEE4 23          inc   hl          Wert nach de
CEE5 56          ld    d,(hl)
CEE6 FE1E        cp    1E          Zeilennummer ?
CEE8 280E        jr    z,CEF8      ja, fertig
CEEA FE1D        cp    1D          Zeilenadresse ?
CEEC C27BD0      jp    nz,D07B     nein, 'Syntax error'
CEEF E5          push  hl
CEF0 EB          ex    de,hl       hl zeigt auf Zeilenbeginn
CEF1 23          inc   hl
CEF2 23          inc   hl
CEF3 23          inc   hl
CEF4 5E          ld    e,(hl)
CEF5 23          inc   hl          Zeilennummer nach de
CEF6 56          ld    d,(hl)
CEF7 E1          pop   hl
CEF8 C33FDD      jp    DD3F        Blanks überlesen

********************************************** Ausdruck holen
CEFB C5          push  bc
CEFC 2B          dec   hl
CEFD 0600        ld    b,00        Hierarchiekode
CEFF CD07CF      call  CF07        Term holen
CF02 C1          pop   bc
CF03 2B          dec   hl
CF04 C33FDD      jp    DD3F        Blanks überlesen

************************************************* Term holen
CF07 C5          push  bc
CF08 CDCBCF      call  CFCB        Term holen
CF0B E5          push  hl
CF0C E1          pop   hl
CF0D C1          pop   bc
CF0E 7E          ld    a,(hl)
CF0F FEEE        cp    EE          '>'
CF11 D8          ret   c           kleiner ?
CF12 FEFE        cp    FE          'NOT'
CF14 D0          ret   nc          größer gleich ?
CF15 FEF4        cp    F4          '+'
CF17 3840        jr    c,CF59      kleiner, dann Vergleichsoperator
CF19 CC45FF      call  z,FF45      '+', dann Test auf String
CF1C 2012        jr    nz,CF30     kein String
CF1E C5          push  bc
CF1F E5          push  hl
```

```
CF20  2AC2B0     ld    hl,(B0C2)    Stringdescriptor
CF23  E3         ex    (sp),hl      auf Stack
CF24  CDCBCF     call  CFCB         n41chsten Term holen
CF27  CD3CFF     call  FF3C         Typ 'String', sonst 'Type mismatch'
CF2A  E3         ex    (sp),hl
CF2B  CD63F8     call  F863         Stringaddition
CF2E  18DC       jr    CF0C         n41chsten Term bearbeiten

****************************************** arithmetische Operatoren
CF30  7E         ld    a,(hl)
CF31  D6F4       sub   F4           minus F4
CF33  87         add   a,a
CF34  87         add   a,a          mal 4
CF35  C681       add   a,81
CF37  5F         ld    e,a          plus CF81, Tabellenadresse
CF38  CECF       adc   a,CF
CF3A  93         sub   e
CF3B  57         ld    d,a
CF3C  EB         ex    de,hl
CF3D  78         ld    a,b
CF3E  BE         cp    (hl)
CF3F  EB         ex    de,hl
CF40  D0         ret   nc
CF41  C5         push  bc
CF42  CD53FF     call  FF53         Ergebnis auf BASIC-Stack ablegen
CF45  D5         push  de
CF46  C5         push  bc
CF47  1A         ld    a,(de)       Hierarchiekode
CF48  47         ld    b,a
CF49  CD07CF     call  CF07         Term holen
CF4C  C1         pop   bc
CF4D  E3         ex    (sp),hl
CF4E  23         inc   hl
CF4F  EB         ex    de,hl
CF50  79         ld    a,c
CF51  CDA0F5     call  F5A0         Platz im BASIC-Stack freigeben
CF54  CDFBFF     call  FFFB         jp (de), Operation ausführen
CF57  18B3       jr    CF0C         n41chsten Term bearbeiten

******************************************** Vergleichsoperatoren
CF59  78         ld    a,b
CF5A  FE0A       cp    0A
CF5C  D0         ret   nc
CF5D  C5         push  bc
CF5E  7E         ld    a,(hl)       Token
CF5F  D6ED       sub   ED           minus Offset
CF61  47         ld    b,a
CF62  CD45FF     call  FF45         Test auf String
CF65  11A9CF     ld    de,CFA9      Adresse für arithmetische Vergleiche
CF68  20D8       jr    nz,CF42      kein String
CF6A  E5         push  hl
CF6B  2AC2B0     ld    hl,(B0C2)    Stringdescriptor
CF6E  E3         ex    (sp),hl      auf Stack
CF6F  C5         push  bc
CF70  060A       ld    b,0A         Hierarchiekode
```

```
CF72  CD07CF        call   CF07          Term holen
CF75  C1            pop    bc
CF76  E3            ex     (sp),hl
CF77  C5            push   bc
CF78  CD97F8        call   F897          Stringvergleich
CF7B  C1            pop    bc
CF7C  CDAFCF        call   CFAF          Ergebnis des Vergleichs holen
CF7F  188B          jr     CF0C          nächsten Term bearbeiten

********************************** BASIC-Operatoren Hierarchiekodes + Adressen
CF81  0C            db     0C            F4,'+'
CF82  C3CCFC        jp     FCCC
CF85  0C            db     0C            F5,'-'
CF86  C3E1FC        jp     FCE1
CF89  12            db     12            F6,'*'
CF8A  C3F5FC        jp     FCF5
CF8D  12            db     12            F7,'/'
CF8E  C312FD        jp     FD12
CF91  16            db     16            F8,'↑'
CF92  C3F4D4        jp     D4F4
CF95  10            db     10            F9,'Backslash'
CF96  C337FD        jp     FD37
CF99  06            db     06            FA,'AND'
CF9A  C358FD        jp     FD58
CF9D  0E            db     0E            FB,'MOD'
CF9E  C349FD        jp     FD49
CFA1  04            db     04            FC,'OR'
CFA2  C363FD        jp     FD63
CFA5  02            db     02            FD,'XOR'
CFA6  C36DFD        jp     FD6D

******************************************* arithmetischer Vergleich
CFA9  0A            ld     a,(bc)
CFAA  C5            push   bc
CFAB  CD09FD        call   FD09          arithmetischer Vergleich
CFAE  C1            pop    bc
CFAF  C601          add    a,01
CFB1  8F            adc    a,a
CFB2  A0            and    b
CFB3  C6FF          add    a,FF
CFB5  9F            sbc    a,a
CFB6  C305FF        jp     FF05          Vorzeichen übernehmen

******************************************* '-' negatives Vorzeichen
CFB9  2B            dec    hl
CFBA  0614          ld     b,14          Hierarchie-Kode
CFBC  CD07CF        call   CF07          Term holen
CFBF  C389FD        jp     FD89          Vorzeichen wechseln

******************************************* BASIC-Operator NOT
CFC2  2B            dec    hl
CFC3  0608          ld     b,08          Hierarchie-Kode
CFC5  CD07CF        call   CF07          Term holen
CFC8  C377FD        jp     FD77          NOT-Operator
```

```
****************************************************** Ausdruck holen
CFCB CD3FDD        call   DD3F          Blanks überlesen

****************************************************** Ausdruck holen
CFCE 281D          jr     z,CFED        'Operand missing'
CFD0 FE0E          cp     0E
CFD2 3839          jr     c,D00D        Variable holen
CFD4 FE20          cp     20
CFD6 3854          jr     c,D02C        numerischen Wert holen
CFD8 FE22          cp     22            '"'
CFDA CACBF7        jp     z,F7CB        zur Stringverarbeitung
CFDD FEFF          cp     FF            Funktion ?
CFDF CA80D0        jp     z,D080        zur Funktionsberechnung

CFE2 E5            push   hl
CFE3 21F2CF        ld     hl,CFF2       Basisadresse der Tabelle
CFE6 CD93FF        call   FF93          Tabelle durchsuchen
CFE9 E3            ex     (sp),hl
CFEA C33FDD        jp     DD3F          Blanks überlesen

CFED 1E16          ld     e,16          'Operand missing'
CFEF C394CA        jp     CA94          Fehlermeldung ausgeben

****************************************************** Sonderfunktionen
CFF2 08            db     08            Anzahl der Tabelleneintr41ge
CFF3 78D0          dw     D078          nicht gefunden, 'Syntax error'
CFF5 F5            db     F5            '-'
CFF6 B9CF          dw     CFB9
CFF8 F4            dw     F4            '+'
CFF9 CECF          dw     CFCE
CFFB 28            db     28            '('
CFFC 70D0          dw     D070
CFFE FE            db     FE            'NOT'
CFFF C2CF          dw     CFC2
D001 E3            db     E3            'ERL'
D002 EED0          dw     D0EE
D004 E4            db     E4            'FN'
D005 30D1          dw     D130
D007 AC            db     AC            'MID$'
D008 4BF9          dw     F94B
D00A 40            db     40            '@'
D00B FAD0          dw     D0FA

****************************************************** Variable holen
D00D CD90D6        call   D690          Variablenadresse holen
D010 300B          jr     nc,D01D       noch nicht angelegt ?
D012 FE03          cp     03            Variablentyp
D014 280F          jr     z,D025        String ?
D016 E5            push   hl
D017 EB            ex     de,hl
D018 CD4BFF        call   FF4B
D01B E1            pop    hl
D01C C9            ret
```

```
D01D  FE03          cp     03          String ?
D01F  C2F3FE        jp     nz,FEF3     Variable löschen
D022  112BD0        ld     de,D02B     Wert auf Null setzen
D025  EB            ex     de,hl
D026  22C2B0        ld     (B0C2),hl
D029  EB            ex     de,hl
D02A  C9            ret

*******************************************************************
D02B  00            db     00          Null

*********************************************************** numerischen Wert holen
D02C  D60E          sub    0E          Offset abziehen
D02E  FE0A          cp     0A          kleiner 10 ?
D030  381D          jr     c,D04F      ja, Ziffer holen
D032  23            inc    hl
D033  FE0B          cp     0B          Ein-Byte-Wert ?
D035  2817          jr     z,D04E
D037  FE0F          cp     0F          Zwei-Byte-Wert (dez, hex, bin) ?
D039  380E          jr     c,D049
D03B  FE11          cp     11
D03D  381A          jr     c,D059      Fließkommawert ?
D03F  203A          jr     nz,D07B     'Syntax error'
D041  3E05          ld     a,05        'Real'
D043  CD4BFF        call   FF4B
D046  2B            dec    hl
D047  1824          jr     D06D        Blanks überlesen

************************************************************ Zwei-Byte-Wert holen
D049  5E            ld     e,(hl)
D04A  23            inc    hl
D04B  56            ld     d,(hl)
D04C  1804          jr     D052        Integerzahl in de übernehmen

************************************************************* Ein-Byte-Wert holen
D04E  7E            ld     a,(hl)
D04F  5F            ld     e,a
D050  1600          ld     d,00        Hi-Byte auf Null setzen
D052  EB            ex     de,hl
D053  CD0DFF        call   FF0D        Intgerzahl in hl übernehmen
D056  EB            ex     de,hl
D057  1814          jr     D06D        Blanks überlesen

*********************************************************** Fließkommawert holen
D059  5E            ld     e,(hl)
D05A  23            inc    hl
D05B  56            ld     d,(hl)
D05C  E5            push   hl
D05D  FE0F          cp     0F
D05F  2007          jr     nz,D068
D061  13            inc    de
D062  EB            ex     de,hl
D063  23            inc    hl
D064  23            inc    hl
D065  5E            ld     e,(hl)
```

```
D066 23          inc   hl
D067 56          ld    d,(hl)
D068 EB          ex    de,hl
D069 CD60FE      call  FE60        Variablentyp auf 'Real'
D06C E1          pop   hl
D06D C33FDD      jp    DD3F        Blanks überlesen

*************************************** '(' Term in Klammern holen
D070 CDFBCE      call  CEFB        Ausdruck holen
D073 CD37DD      call  DD37        Test auf nachfolgendes Zeichen
D076 29          db    29          ')'
D077 C9          ret

**************************************************
D078 CD0DAC      call  AC0D        ret
D07B 1E02        ld    e,02        'Syntax error'
D07D C394CA      jp    CA94        Fehlermeldung ausgeben

*************************************** Funktionsberechnung
D080 23          inc   hl          Programmzeiger erhöhen
D081 4E          ld    c,(hl)      Token holen
D082 CD3FDD      call  DD3F        Blanks überlesen
D085 79          ld    a,c         Token testen
D086 FE40        cp    40
D088 3805        jr    c,D08F      40 - 48, reservierte Variable
D08A FE49        cp    49
D08C DABBD0      jp    c,D0BB
D08F CD37DD      call  DD37        Test auf nachfolgendes Zeichen
D092 28          db    28          '('
D093 79          ld    a,c
D094 87          add   a,a
D095 C61E        add   a,1E
D097 4F          ld    c,a
D098 FE59        cp    59
D09A 300D        jr    nc,D0A9     'Syntax error'
D09C FE1D        cp    1D
D09E 380E        jr    c,D0AE      Funktion berechnen
D0A0 CD70D0      call  D070        Funktionsargument in Klammern holen
D0A3 E5          push  hl
D0A4 CDAED0      call  D0AE        Funktion berechnen
D0A7 E1          pop   hl
D0A8 C9          ret

D0A9 CD0AAC      call  AC0A        ret
D0AC 18CD        jr    D07B        'Syntax error'

D0AE E5          push  hl
D0AF 0600        ld    b,00
D0B1 2190D1      ld    hl,D190     Adressen der Funktionen
D0B4 09          add   hl,bc
D0B5 7E          ld    a,(hl)
D0B6 23          inc   hl
D0B7 66          ld    h,(hl)
D0B8 6F          ld    l,a
D0B9 E3          ex    (sp),hl
```

```
D0BA C9          ret

D0BB E5          push  hl
D0BC 4F          ld    c,a
D0BD 0600        ld    b,00
D0BF 214AD0      ld    hl,D04A
D0C2 09          add   hl,bc
D0C3 09          add   hl,bc
D0C4 7E          ld    a,(hl)
D0C5 23          inc   hl
D0C6 66          ld    h,(hl)
D0C7 6F          ld    l,a
D0C8 E3          ex    (sp),hl
D0C9 C9          ret

**************************************** Adressen der reservierten Variablen
D0CA 17C4        dw    C417           40, EOF
D0CC DCD0        dw    D0DC           41, ERR
D0CE F4D0        dw    D0F4           42, HIMEM
D0D0 24FA        dw    FA24           43, INKEY$
D0D2 DBD4        dw    D4DB           44, PI
D0D4 84D5        dw    D584           45, RND
D0D6 E5D0        dw    D0E5           46, TIME
D0D8 07D1        dw    D107           47, XPOS
D0DA 0ED1        dw    D10E           48, YPOS

************************************************************ ERR
D0DC E5          push  hl
D0DD 3AAAAD      ld    a,(ADAA)       ERROR-Nummer
D0E0 CD0AFF      call  FF0A           Akkuinhalt als Integerzahl übernehmen
D0E3 E1          pop   hl
D0E4 C9          ret

*********************************************************** TIME
D0E5 E5          push  hl
D0E6 CD0DBD      call  BD0D           KL TIME PLEASE
D0E9 CD7CFE      call  FE7C           4-Byte-Wert nach Fließkomma wandeln
D0EC E1          pop   hl
D0ED C9          ret

************************************************************ ERL
D0EE E5          push  hl
D0EF CDDFCA      call  CADF           ERROR-Zeilennummer holen
D0F2 180E        jr    D102

********************************************************** HIMEM
D0F4 E5          push  hl
D0F5 2A7BAE      ld    hl,(AE7B)      HIMEM
D0F8 1808        jr    D102

********************************************* '@', Variablenpointer
D0FA CD90D6      call  D690           Variablenadresse holen
D0FD D2ABCE      jp    nc,CEAB        nicht vorhanden, 'Improper argument'
D100 E5          push  hl
D101 EB          ex    de,hl
```

```
D102  CD60FE     call   FE60        Wert übernehmen
D105  E1         pop    hl
D106  C9         ret

************************************************************ XPOS
D107  E5         push   hl
D108  CDC6BB     call   BBC6        GRA ASK CURSOR
D10B  EB         ex     de,hl
D10C  1804       jr     D112

************************************************************ YPOS
D10E  E5         push   hl
D10F  CDC6BB     call   BBC6        GRA ASK CURSOR
D112  CD0DFF     call   FF0D        Integerzahl in hl übernehmen
D115  E1         pop    hl
D116  C9         ret

****************************************************** BASIC-Befehl DEF
D117  CD37DD     call   DD37        Test auf nachfolgendes Zeichen
D11A  E4         db     E4          'FN'
D11B  EB         ex     de,hl
D11C  CDD6DD     call   DDD6        Zeilennummer nach hl holen
D11F  EB         ex     de,hl
D120  1E0C       ld     e,0C        'Invalid direct command'
D122  D294CA     jp     nc,CA94     Direktmodus, Fehlermeldung ausgeben

D125  CDA2D6     call   D6A2        Funktion suchen
D128  EB         ex     de,hl
D129  73         ld     (hl),e
D12A  23         inc    hl
D12B  72         ld     (hl),d
D12C  EB         ex     de,hl
D12D  C3EFE8     jp     E8EF        Rest des Statements überlesen

****************************************************** BASIC-Funktion FN
D130  CDA2D6     call   D6A2        Funktion suchen
D133  C5         push   bc
D134  E5         push   hl
D135  EB         ex     de,hl
D136  5E         ld     e,(hl)
D137  23         inc    hl
D138  56         ld     d,(hl)
D139  EB         ex     de,hl
D13A  7C         ld     a,h
D13B  B5         or     l
D13C  1E12       ld     e,12        'Unknown user function'
D13E  CA94CA     jp     z,CA94      Fehlermeldung ausgeben

D141  CD07DA     call   DA07
D144  7E         ld     a,(hl)
D145  FE28       cp     28          '('
D147  202C       jr     nz,D175
D149  CD3FDD     call   DD3F        Blanks überlesen
D14C  E3         ex     (sp),hl
D14D  CD37DD     call   DD37        Test auf nachfolgendes Zeichen
```

```
D150  28            db    28          '('
D151  E3            ex    (sp),hl
D152  CD4BDA        call  DA4B
D155  E3            ex    (sp),hl
D156  D5            push  de
D157  CDFBCE        call  CEFB        Ausdruck holen
D15A  E3            ex    (sp),hl
D15B  78            ld    a,b
D15C  CD66D6        call  D666
D15F  E1            pop   hl
D160  CD55DD        call  DD55        folgt Komma ?
D163  3007          jr    nc,D16C     nein
D165  E3            ex    (sp),hl
D166  CD37DD        call  DD37        Test auf nachfolgendes Zeichen
D169  2C            db    2C          ','
D16A  18E6          jr    D152

D16C  CD37DD        call  DD37        Test auf nachfolgendes Zeichen
D16F  29            db    29          ')'
D170  E3            ex    (sp),hl
D171  CD37DD        call  DD37        Test auf nachfolgendes Zeichen
D174  29            db    29          ')'
D175  CD27DA        call  DA27
D178  CD37DD        call  DD37        Test auf nachfolgendes Zeichen
D17B  EF            db    EF          '='
D17C  CDFBCE        call  CEFB        Ausdruck holen
D17F  C27BD0        jp    nz,D07B     'Syntax error'
D182  CD30DA        call  DA30
D185  CD45FF        call  FF45        Test auf String
D188  CC49FB        call  z,FB49      ja
D18B  E1            pop   hl
D18C  F1            pop   af
D18D  C3D7FE        jp    FED7

********************************* BASIC-Funktionen mit mehreren Argumenten
D190  BAF8          dw    F8BA        71, BIN$
D192  EAF8          dw    F8EA        72, DEC$
D194  C4F8          dw    F8C4        73, HEX$
D196  A1FA          dw    FAA1        74, INSTR
D198  3CF9          dw    F93C        75, LEFT$
D19A  EED1          dw    D1EE        76, MAX
D19C  EAD1          dw    D1EA        77, MIN
D19E  76C2          dw    C276        78, POS
D1A0  43F9          dw    F943        79, RIGHT$
D1A2  19D2          dw    D219        7A, ROUND
D1A4  36FA          dw    FA36        7B, STRING$
D1A6  E9C4          dw    C4E9        7C, TEST
D1A8  EEC4          dw    C4EE        7D, TESTR
D1AA  ABCE          dw    CEAB        7E, 'Improper argument'
D1AC  62C2          dw    C262        7F, VPOS

************************************** Adressen der BASIC-Funktionen
D1AE  85FD          dw    FD85        00, ABS
D1B0  10FA          dw    FA10        01, ASC
D1B2  3ED5          dw    D53E        02, ATN
```

```
D1B4  16FA      dw    FA16       03, CHR$
D1B6  8DFE      dw    FE8D       04, CINT
D1B8  34D5      dw    D534       05, COS
D1BA  ECFE      dw    FEEC       06, CREAL
D1BC  20D5      dw    D520       07, EXP
D1BE  E8FD      dw    FDE8       08, FIX
D1C0  2DFC      dw    FC2D       09, FRE
D1C2  09D4      dw    D409       0A, INKEY
D1C4  6DF1      dw    F16D       0B, INP
D1C6  EDFD      dw    FDED       0C, INT
D1C8  23D4      dw    D423       0D, JOY
D1CA  0AFA      dw    FA0A       0E, LEN
D1CC  2AD5      dw    D52A       0F, LOG
D1CE  25D5      dw    D525       10, LOG10
D1D0  34F8      dw    F834       11, LOWER$
D1D2  58F1      dw    F158       12, PEEK
D1D4  9FC9      dw    C99F       13, REMAIN
D1D6  02FF      dw    FF02       14, SGN
D1D8  2FD5      dw    D52F       15, SIN
D1DA  57FA      dw    FA57       16, SPACE$
D1DC  29D3      dw    D329       17, SQ
D1DE  EFD4      dw    D4EF       18, SQR
D1E0  1EF9      dw    F91E       19, STR$
D1E2  39D5      dw    D539       1A, TAN
D1E4  C2FE      dw    FEC2       1B, UNT
D1E6  42F8      dw    F842       1C, UPPER$
D1E8  77FA      dw    FA77       1D, VAL

*********************************************** BASIC-Funktion MIN
D1EA  06FF      ld    b,FF       Flag für MIN
D1EC  1802      jr    D1F0

*********************************************** BASIC-Funktion MAX
D1EE  0601      ld    b,01       Flag für MAX
D1F0  CDFBCE    call  CEFB       Ausdruck holen
D1F3  CD55DD    call  DD55       folgt Komma ?
D1F6  301C      jr    nc,D214    nein, fertig
D1F8  CD53FF    call  FF53       Variable auf BASIC-Stack ablegen
D1FB  CDFBCE    call  CEFB       Ausdruck holen
D1FE  E5        push  hl
D1FF  79        ld    a,c
D200  CDA0F5    call  F5A0       Platz im BASIC-Stack freigeben
D203  C5        push  bc
D204  E5        push  hl
D205  CD09FD    call  FD09       arithmetischer Vergleich
D208  E1        pop   hl
D209  C1        pop   bc
D20A  B7        or    a
D20B  2804      jr    z,D211
D20D  B8        cp    b
D20E  C44EFF    call  nz,FF4E    Ergebnis des Vergleichs holen
D211  E1        pop   hl
D212  18DF      jr    D1F3       nächstes Argument
```

```
D214  CD37DD     call   DD37         Test auf nachfolgendes Zeichen
D217  29         db     29           ')'
D218  C9         ret

********************************************* BASIC-Funktion ROUND
D219  CDFBCE     call   CEFB         Ausdruck holen
D21C  CD53FF     call   FF53         und auf BASIC-Stack ablegen
D21F  CD55DD     call   DD55         folgt Komma ?
D222  110000     ld     de,0000      Defaultwert 0
D225  DC86CE     call   c,CE86       ja, Integerwert mit Vorzeichen holen
D228  CD37DD     call   DD37         Test auf nachfolgendes Zeichen
D22B  29         db     29           ')'
D22C  E5         push   hl
D22D  D5         push   de
D22E  212700     ld     hl,0027      39
D231  19         add    hl,de        addieren
D232  114F00     ld     de,004F      79
D235  CDB8FF     call   FFB8         Vergleich hl <> de
D238  D2ABCE     jp     nc,CEAB      größer, 'Improper argument'
D23B  D1         pop    de
D23C  79         ld     a,c
D23D  CDA0F5     call   F5A0         Platz im BASIC-Stack freigeben
D240  43         ld     b,e          Rundungstelllenzahl nach b
D241  CDAFFD     call   FDAF         Zahl runden
D244  E1         pop    hl
D245  C9         ret

************************************************ BASIC-Befehl CAT
D246  C0         ret    nz
D247  E5         push   hl
D248  CDADD2     call   D2AD         Kassetten-I/O abbrechen
D24B  CD37F6     call   F637         Ausgabepuffer anlegen
D24E  CD9BBC     call   BC9B         CAS CATALOG
D251  CD71F6     call   F671         Ausgabepuffer freigeben
D254  E1         pop    hl
D255  C9         ret

********************************************* BASIC-Befehl OPENOUT
D256  CD73D2     call   D273         Filename holen
D259  CD37F6     call   F637         Ausgabepuffer anlegen
D25C  C38CBC     jp     BC8C         CAS OUT OPEN

********************************************** BASIC-Befehl OPENIN
D25F  CD6AD2     call   D26A         Namen holen, File öffnen
D262  FE16       cp     16           ASCII-File ?
D264  C8         ret    z
D265  1E19       ld     e,19         'File type error'
D267  C394CA     jp     CA94         Fehlermeldung ausgeben

****************************************** Namen holen, Eingabefile öffnen
D26A  CD73D2     call   D273         Filename holen
D26D  CD32F6     call   F632         Eingabepuffer anlegen
D270  C377BC     jp     BC77         CAS IN OPEN
```

```
****************************************************************
D273  CD9FCE     call  CE9F       Stringausdruck und -Parameter holen
D276  E3         ex    (sp),hl
D277  EB         ex    de,hl
D278  CD85D2     call  D285       Test auf Systemmeldungen
D27B  CA6BCB     jp    z,CB6B     Abbruch durch 'ESC'
D27E  E1         pop   hl
D27F  D8         ret   c
D280  1E1B       ld    e,1B       'File already open'
D282  C394CA     jp    CA94       Fehlermeldung ausgeben

****************************************************************
D285  D5         push  de
D286  0E00       ld    c,00       Flag für Meldungen ausgeben
D288  78         ld    a,b        Länge des Filenamens
D289  B7         or    a
D28A  2808       jr    z,D294     kein Filename ?
D28C  7E         ld    a,(hl)     erstes Zeichen
D28D  FE21       cp    21         '!'
D28F  2003       jr    nz,D294    nein
D291  23         inc   hl         Zeiger auf zweites Zeichen setzen
D292  05         dec   b          Länge erniedrigen
D293  0D         dec   c          Flag für Meldungen unterdrücken
D294  79         ld    a,c
D295  C36BBC     jp    BC6B       CAS NOISY

****************************************** BASIC-Befehl CLOSEIN
D298  E5         push  hl
D299  CD7ABC     call  BC7A       CAS IN CLOSE
D29C  CD6DF6     call  F66D       Eingabepuffer freigeben
D29F  E1         pop   hl
D2A0  C9         ret

***************************************** BASIC-Befehl CLOSEOUT
D2A1  E5         push  hl
D2A2  CD8FBC     call  BC8F       CAS OUT CLOSE
D2A5  CA6BCB     jp    z,CB6B     'Break in Zeilennummer'
D2A8  CD71F6     call  F671       Ausgabepuffer freigeben
D2AB  E1         pop   hl
D2AC  C9         ret

***************************************** Kassetten-I/O abbrechen
D2AD  C5         push  bc
D2AE  D5         push  de
D2AF  E5         push  hl
D2B0  CD7DBC     call  BC7D       CAS IN ABANDON
D2B3  CD6DF6     call  F66D       Eingabepuffer freigeben
D2B6  CD92BC     call  BC92       CAS OUT ABANDON
D2B9  CD71F6     call  F671       Ausgabepuffer freigeben
D2BC  E1         pop   hl
D2BD  D1         pop   de
D2BE  C1         pop   bc
D2BF  C9         ret
```

```
*************************************************** BASIC-Befehl SOUND
D2C0 CD67CE      call  CE67        8-Bit-Wert holen
D2C3 32B2AD      ld    (ADB2),a    Kanal-Status
D2C6 CD37DD      call  DD37        Test auf nachfolgendes Zeichen
D2C9 2C          db    2C          ','
D2CA CDFFD3      call  D3FF        Argument 0 bis 4095 holen
D2CD ED53B5AD    ld    (ADB5),de   Ton-Periode
D2D1 CD55DD      call  DD55        folgt Komma ?
D2D4 111400      ld    de,0014     Default-Wert 20
D2D7 DC86CE      call  c,CE86      ja, Integerwert mit Vorzeichen holen
D2DA ED53B9AD    ld    (ADB9),de   Dauer
D2DE 010C10      ld    bc,100C     max. 15, Default 12
D2E1 CD0DD3      call  D30D        falls vorhanden Argument holen
D2E4 32B8AD      ld    (ADB8),a    Lautstärke
D2E7 0E00        ld    c,00        max. 15, Default 0
D2E9 CD0DD3      call  D30D        falls vorhanden Argument holen
D2EC 32B3AD      ld    (ADB3),a    Lautstärken-Hüllkurve
D2EF CD0DD3      call  D30D        falls vorhanden Argument holen
D2F2 32B4AD      ld    (ADB4),a    Ton-Hüllkurve
D2F5 0620        ld    b,20        max. 31, Default 0
D2F7 CD0DD3      call  D30D        falls vorhanden Argument holen
D2FA 32B7AD      ld    (ADB7),a    Geräusch-Periode
D2FD CD4ADD      call  DD4A        Ende des Statements, sonst 'Syntax error'
D300 E5          push  hl
D301 21B2AD      ld    hl,ADB2     Adresse des Sound-Parameterblocks
D304 CDAABC      call  BCAA        SOUND QUEUE
D307 E1          pop   hl
D308 D8          ret   c
D309 F1          pop   af
D30A C371DD      jp    DD71        zur Interpreterschleife

*************************************** falls vorhanden 8-Bit-Wert holen
D30D CD55DD      call  DD55        folgt Komma ?
D310 79          ld    a,c         Default-Wert laden
D311 D0          ret   nc          kein Komma, fertig
D312 7E          ld    a,(hl)
D313 FE2C        cp    2C          ','
D315 79          ld    a,c
D316 C8          ret   z
D317 CD67CE      call  CE67        8-Bit-Wert holen
D31A B8          cp    b           größer gleich b ?
D31B D8          ret   c           nein
D31C 182B        jr    D349        'Improper argument'

************************************************* BASIC-Befehl RELEASE
D31E 0608        ld    b,08        8
D320 CD17D3      call  D317        8-Bit-Wert < 8 holen
D323 E5          push  hl
D324 CDB3BC      call  BCB3        SOUND RELEASE
D327 E1          pop   hl
D328 C9          ret
```

```
********************************************** BASIC-Funktion SQ
D329  CD8DFE     call  FE8D       CINT
D32C  7D         ld    a,l
D32D  B7         or    a
D32E  1F         rra
D32F  3806       jr    c,D337
D331  1F         rra
D332  3803       jr    c,D337
D334  1F         rra
D335  3012       jr    nc,D349    'Improper argument'
D337  B4         or    h
D338  200F       jr    nz,D349    'Improper argument'
D33A  7D         ld    a,l
D33B  CDADBC     call  BCAD       SOUND CHECK
D33E  C30AFF     jp    FF0A       Akkuinhalt als Integerzahl übernehmen

**************************************** Argument -128 bis +127 holen
D341  CD86CE     call  CE86       Integerwert mit Vorzeichen holen
D344  7B         ld    a,e
D345  87         add   a,a
D346  9F         sbc   a,a
D347  BA         cp    d
D348  C8         ret   z
D349  1E05       ld    e,05       'Improper argument'
D34B  C394CA     jp    CA94       Fehlermeldung ausgeben

********************************************** BASIC-Befehl ENV
D34E  CD6DCE     call  CE6D       8-Bit-Wert ungleich Null holen
D351  FE10       cp    10         größer gleich 16 ?
D353  30F4       jr    nc,D349    'Improper argument'
D355  F5         push  af
D356  1167D3     ld    de,D367
D359  CDD8D3     call  D3D8
D35C  F1         pop   af
D35D  E5         push  hl
D35E  21BBAD     ld    hl,ADBB
D361  71         ld    (hl),c
D362  CDBCBC     call  BCBC       SOUND AMPL ENVELOPE
D365  E1         pop   hl
D366  C9         ret

D367  7E         ld    a,(hl)
D368  FEEF       cp    EF         '='
D36A  2012       jr    nz,D37E
D36C  CD3FDD     call  DD3F       Blanks überlesen
D36F  0610       ld    b,10       16
D371  CD17D3     call  D317       8-Bit-Wert < 16 holen
D374  F680       or    80         Bit 7 setzen
D376  4F         ld    c,a
D377  CD37DD     call  DD37       Test auf nachfolgendes Zeichen
D37A  2C         db    2C         ','
D37B  C391CE     jp    CE91       16-Bit-Wert holen
```

```
D37E  0680      ld    b,80        128
D380  CD17D3    call  D317        8-Bit-Wert < 128 holen
D383  1840      jr    D3C5

********************************************* BASIC-Befehl ENT
D385  CD41D3    call  D341        Argument -128 bis +127 holen
D388  7A        ld    a,d
D389  B7        or    a
D38A  7B        ld    a,e
D38B  2802      jr    z,D38F      Null ?
D38D  2F        cpl   a
D38E  3C        inc   a
D38F  5F        ld    e,a
D390  B7        or    a           Null ?
D391  28B6      jr    z,D349      'Improper argument'
D393  FE10      cp    10          größer gleich 16 ?
D395  30B2      jr    nc,D349     'Improper argument'
D397  D5        push  de
D398  11AED3    ld    de,D3AE
D39B  CDD8D3    call  D3D8
D39E  D1        pop   de
D39F  E5        push  hl
D3A0  21BBAD    ld    hl,ADBB
D3A3  7A        ld    a,d
D3A4  E680      and   80
D3A6  B1        or    c
D3A7  77        ld    (hl),a
D3A8  7B        ld    a,e
D3A9  CDBFBC    call  BCBF        SOUND TONE ENVELOPE
D3AC  E1        pop   hl
D3AD  C9        ret

*******************************************************
D3AE  7E        ld    a,(hl)
D3AF  FEEF      cp    EF          '='
D3B1  200D      jr    nz,D3C0
D3B3  CD3FDD    call  DD3F        Blanks überlesen
D3B6  CDFFD3    call  D3FF        Wert von 0 bis 4095 holen
D3B9  7A        ld    a,d
D3BA  C6F0      add   a,F0
D3BC  4F        ld    c,a
D3BD  43        ld    b,e
D3BE  180E      jr    D3CE

D3C0  06F0      ld    b,F0        240
D3C2  CD17D3    call  D317        8-Bit-Wert < 240 holen
D3C5  4F        ld    c,a
D3C6  CD37DD    call  DD37        Test auf nachfolgendes Zeichen
D3C9  2C        db    2C          ','
D3CA  CD41D3    call  D341        Argument -128 bis +127 holen
D3CD  43        ld    b,e
D3CE  CD37DD    call  DD37        Test auf nachfolgendes Zeichen
D3D1  2C        db    2C          ','
D3D2  CD67CE    call  CE67        8-Bit-Wert holen
D3D5  57        ld    d,a
```

```
D3D6  58              ld     e,b
D3D7  C9              ret

D3D8  010005          ld     bc,0500
D3DB  CD55DD          call   DD55          auf Komma prüfen
D3DE  301C            jr     nc,D3FC
D3E0  D5              push   de
D3E1  C5              push   bc
D3E2  CDFBFF          call   FFFB          jp (de)
D3E5  79              ld     a,c
D3E6  C1              pop    bc
D3E7  C5              push   bc
D3E8  E5              push   hl
D3E9  21BCAD          ld     hl,ADBC
D3EC  0600            ld     b,00
D3EE  09              add    hl,bc
D3EF  09              add    hl,bc
D3F0  09              add    hl,bc
D3F1  77              ld     (hl),a
D3F2  23              inc    hl
D3F3  73              ld     (hl),e
D3F4  23              inc    hl
D3F5  72              ld     (hl),d
D3F6  E1              pop    hl
D3F7  C1              pop    bc
D3F8  0C              inc    c
D3F9  D1              pop    de
D3FA  10DF            djnz   D3DB
D3FC  C34ADD          jp     DD4A          Ende des Statements, sonst 'Syntax error'

*************************************** Argument 0 bis 4095 holen
D3FF  CD86CE          call   CE86          Integerwert mit Vorzeichen holen
D402  7A              ld     a,d           Hi-Byte
D403  E6F0            and    F0            Bit 12-15 gesetzt ?
D405  C249D3          jp     nz,D349       ja, 'Improper argument'
D408  C9              ret

*************************************** BASIC-Funktion INKEY
D409  CD8DFE          call   FE8D          CINT
D40C  115000          ld     de,0050       80
D40F  CDB8FF          call   FFB8          Vergleich hl <> de
D412  3022            jr     nc,D436       'Improper argument'
D414  7D              ld     a,l
D415  CD1EBB          call   BB1E          KM TEST KEY
D418  21FFFF          ld     hl,FFFF       -1, wenn nicht gedrückt
D41B  2803            jr     z,D420
D41D  69              ld     l,c
D41E  2600            ld     h,00
D420  C30DFF          jp     FF0D          Integerzahl in hl übernehmen

*************************************** BASIC-Funktion JOY
D423  CD24BB          call   BB24          KM GET JOYSTICK
D426  EB              ex     de,hl
D427  CD8DFE          call   FE8D          CINT
D42A  7C              ld     a,h
```

```
D42B  B5          or    l
D42C  2802        jr    z,D430
D42E  53          ld    d,e
D42F  2B          dec   hl
D430  7C          ld    a,h
D431  B5          or    l
D432  7A          ld    a,d
D433  CA0AFF      jp    z,FF0A       Akkuinhalt als Integerzahl übernehmen
D436  C349D3      jp    D349         'Improper argument'

******************************************************** BASIC-Befehl KEY
D439  FE8D        cp    8D           'DEF'
D43B  2819        jr    z,D456
D43D  3E20        ld    a,20
D43F  CD17D3      call  D317         Argument holen, 8-Bit-Wert
D442  F5          push  af           Tastennummer merken
D443  CD37DD      call  DD37         Test auf nachfolgendes Zeichen
D446  2C          db    2C           ','
D447  CD9FCE      call  CE9F         Stringausdruck und -Parameter holen
D44A  48          ld    c,b          Stringlänge nach c
D44B  F1          pop   af
D44C  47          ld    b,a          Tastennummer nach b
D44D  E5          push  hl
D44E  EB          ex    de,hl        Stringadresse nach hl
D44F  CD0FBB      call  BB0F         KM SET EXPAND
D452  E1          pop   hl
D453  30E1        jr    nc,D436      'Improper argument'
D455  C9          ret

****************************************************************** KEY DEF
D456  CD3FDD      call  DD3F         Banks überlesen
D459  CD67CE      call  CE67         8-Bit-Wert holen, Tastennnummer
D45C  4F          ld    c,a
D45D  FE50        cp    50           80
D45F  30D5        jr    nc,D436      größer gleich, 'Improper argument'
D461  CD37DD      call  DD37         Test auf nachfolgendes Zeichen
D464  2C          db    2C           ','
D465  0602        ld    b,02         2
D467  CD17D3      call  D317         Argument < 2 holen
D46A  1F          rra
D46B  9F          sbc   a,a
D46C  47          ld    b,a
D46D  C5          push  bc
D46E  E5          push  hl
D46F  79          ld    a,c
D470  CD39BB      call  BB39         KM SET REPEAT
D473  E1          pop   hl
D474  C1          pop   bc
D475  1127BB      ld    de,BB27      KM SET TRANSLATE
D478  CD84D4      call  D484         auf weiteres Argument testen
D47B  112DBB      ld    de,BB2D      KM SET SHIFT
D47E  CD84D4      call  D484         auf weiteres Argument testen
D481  1133BB      ld    de,BB33      KM SET CONTROL
D484  CD55DD      call  DD55         folgt Komma ?
D487  D0          ret   nc           nein, fertig
```

```
D488  D5        push  de
D489  CD67CE    call  CE67        8-Bit-Wert holen
D48C  47        ld    b,a
D48D  E3        ex    (sp),hl
D48E  79        ld    a,c
D48F  CDF8FF    call  FFF8        jp (hl)
D492  E1        pop   hl
D493  C9        ret

************************************************** BASIC-Befehl SPEED
D494  FEA4      cp    A4          'KEY'
D496  013FBB    ld    bc,BB3F     KM SET DELAY
D499  2810      jr    z,D4AB
D49B  FEA2      cp    A2          'INK'
D49D  013EBC    ld    bc,BC3E     SCR SET FLASHING
D4A0  2809      jr    z,D4AB
D4A2  FED9      cp    D9          'WRITE'
D4A4  281D      jr    z,D4C3
D4A6  1E02      ld    e,02        'Syntax error'
D4A8  C394CA    jp    CA94        Fehlermeldung ausgeben

************************************************** SPEED KEY & INK
D4AB  C5        push  bc
D4AC  CD3FDD    call  DD3F        Blanks überlesen
D4AF  CD6DCE    call  CE6D        8-Bit-Wert ungleich Null holen
D4B2  4F        ld    c,a
D4B3  CD37DD    call  DD37        Test auf nachfolgendes Zeichen
D4B6  2C        db    2C          ','
D4B7  CD6DCE    call  CE6D        8-Bit-Wert ungleich Null holen
D4BA  5F        ld    e,a
D4BB  51        ld    d,c
D4BC  C1        pop   bc
D4BD  EB        ex    de,hl
D4BE  CDF9FF    call  FFF9        jp (bc)
D4C1  EB        ex    de,hl
D4C2  C9        ret

************************************************** SPEED WRITE
D4C3  CD3FDD    call  DD3F        Blanks überlesen
D4C6  0602      ld    b,02
D4C8  CD17D3    call  D317        Argument < 2 holen
D4CB  E5        push  hl
D4CC  21A700    ld    hl,00A7     167
D4CF  3D        dec   a
D4D0  3E32      ld    a,32
D4D2  2802      jr    z,D4D6      Null ?
D4D4  29        add   hl,hl       nein, Zeitkonstante verdoppeln
D4D5  0F        rrca
D4D6  CD68BC    call  BC68        CAS SET SPEED
D4D9  E1        pop   hl
D4DA  C9        ret
```

```
******************************************************************** PI
D4DB  E5          push  hl
D4DC  CD19FF      call  FF19        Typ auf 'Real' setzen
D4DF  CD1DFF      call  FF1D        Variablentyp nach c, hl auf Variable
D4E2  CD76BD      call  BD76        π holen
D4E5  E1          pop   hl
D4E6  C9          ret

*********************************************** BASIC-Befehl DEG
D4E7  3EFF        ld    a,FF        FF = DEG
D4E9  1801        jr    D4EC

*********************************************** BASIC-Befehl RAD
D4EB  AF          xor   a           0 = RAD
D4EC  C373BD      jp    BD73        DEG/RAD-Modus setzen

********************************************** BASIC-Funktion SQR
D4EF  0179BD      ld    bc,BD79     SQR-Funktion
D4F2  1816        jr    D50A

*********************************************** BASIC-Operator '↑'
D4F4  E5          push  hl
D4F5  C5          push  bc
D4F6  CDECFE      call  FEEC        CREAL
D4F9  EB          ex    de,hl
D4FA  21CBAD      ld    hl,ADCB     Zwischenspeicher für Fließkommazahl
D4FD  CD3DBD      call  BD3D        Variable von (de) nach (hl) kopieren
D500  C1          pop   bc
D501  E3          ex    (sp),hl
D502  79          ld    a,c
D503  CD4BFF      call  FF4B
D506  D1          pop   de
D507  017CBD      ld    bc,BD7C     Potenzierung
D50A  CD19D5      call  D519        Funktion ausführen
D50D  D8          ret   c           fehlerfrei ?
D50E  CAEACA      jp    z,CAEA      'Division by zero'
D511  FAF3CA      jp    m,CAF3      'Overflow'
D514  1E05        ld    e,05        'Improper argument'
D516  C394CA      jp    CA94        Fehlermeldung ausgeben

************************************************************************
D519  C5          push  bc
D51A  D5          push  de
D51B  CDECFE      call  FEEC        CREAL
D51E  D1          pop   de
D51F  C9          ret               Funktion ausführen

********************************************** BASIC-Funktion EXP
D520  0185BD      ld    bc,BD85     EXP-Funktion
D523  18E5        jr    D50A

******************************************** BASIC-Funktion LOG10
D525  0182BD      ld    bc,BD82     LOG10-Funktion
D528  18E0        jr    D50A
```

```
********************************************** BASIC-Funktion LOG
D52A 017FBD       ld     bc,BD7F      LOG-Funktion
D52D 18DB         jr     D50A

********************************************** BASIC-Funktion SIN
D52F 0188BD       ld     bc,BD88      SIN-Funktion
D532 18D6         jr     D50A

********************************************** BASIC-Funktion COS
D534 018BBD       ld     bc,BD8B      COS-Funktion
D537 18D1         jr     D50A

********************************************** BASIC-Funktion TAN
D539 018EBD       ld     bc,BD8E      TAN-Funktion
D53C 18CC         jr     D50A

********************************************** BASIC-Funktion ATN
D53E 0191BD       ld     bc,BD91      ATN-Funktion
D541 18C7         jr     D50A

*********************************************************
D543 52 61 6E 64 6F 6D 20 6E          'Random number seed ?'
D54B 75 6D 62 65 72 20 73 65
D553 65 64 20 3F 20 00

******************************************** BASIC-Befehl RANDOMIZE
D559 2806         jr     z,D561
D55B CDFBCE       call   CEFB         Ausdruck holen
D55E E5           push   hl
D55F 181B         jr     D57C

D561 E5           push   hl
D562 2143D5       ld     hl,D543      'Random number seed ?'
D565 CD41C3       call   C341         ausgeben
D568 CD3BCA       call   CA3B         Eingabezeile holen
D56B D26BCB       jp     nc,CB6B      ESC-Taste gedrückt ?
D56E CD4EC3       call   C34E         LF ausgeben
D571 CDA3EC       call   ECA3         Eingabe lesen
D574 30EC         jr     nc,D562      ungültig, wiederholen
D576 CD61DD       call   DD61         Blank, TAB und LF überlesen
D579 B7           or     a
D57A 20E6         jr     nz,D562
D57C CDECFE       call   FEEC         CREAL
D57F CD9ABD       call   BD9A         Set Random Seed
D582 E1           pop    hl
D583 C9           ret

****************************************************** RND
D584 7E           ld     a,(hl)
D585 FE28         cp     28           '('
D587 201C         jr     nz,D5A5
D589 CD3FDD       call   DD3F         Blanks überlesen
D58C CDFBCE       call   CEFB         Ausdruck holen
D58F CD37DD       call   DD37         Test auf nachfolgendes Zeichen
D592 29           db     29           ')'
```

```
D593  E5        push  hl
D594  CDECFE    call  FEEC        CREAL
D597  CD70BD    call  BD70        SGN
D59A  2005      jr    nz,D5A1     ungleich Null ?
D59C  CDA0BD    call  BDA0        letzten RND-Wert holen
D59F  E1        pop   hl
D5A0  C9        ret

D5A1  FC9ABD    call  m,BD9A      Set Random Seed
D5A4  E1        pop   hl
D5A5  E5        push  hl
D5A6  CD16FF    call  FF16        Typ auf Fließkomma setzen
D5A9  CD9DBD    call  BD9D        RND
D5AC  E1        pop   hl
D5AD  C9        ret

****************************************** Variablenzeiger rücksetzen
D5AE  CDBED5    call  D5BE        Tabelle löschen
D5B1  2A83AE    ld    hl,(AE83)   Programmende
D5B4  2285AE    ld    (AE85),hl   Variablenstart
D5B7  2287AE    ld    (AE87),hl   Arraystart
D5BA  2289AE    ld    (AE89),hl   Arrayende
D5BD  C9        ret

********************************************** Tabellen löschen
D5BE  21D0AD    ld    hl,ADD0
D5C1  3E36      ld    a,36        54 = 2*27, Variablen + Funktionen
D5C3  CDCBD5    call  D5CB        ADD0 bis AE05 löschen
D5C6  2106AE    ld    hl,AE06
D5C9  3E06      ld    a,06        AE06 bis AE0B löschen
D5CB  3600      ld    (hl),00     Arrays
D5CD  23        inc   hl
D5CE  3D        dec   a
D5CF  20FA      jr    nz,D5CB
D5D1  C9        ret

********************************************** Flag für FN löschen
D5D2  210000    ld    hl,0000
D5D5  2204AE    ld    (AE04),hl
D5D8  C9        ret

****************************************** Tabellenadresse berechnen
D5D9  3E5B      ld    a,5B        'Z'+1, FN
D5DB  2A85AE    ld    hl,(AE85)   Variablenstart
D5DE  2B        dec   hl          minus 1
D5DF  44        ld    b,h         nach bc
D5E0  4D        ld    c,l
D5E1  87        add   a,a         mal 2
D5E2  C64E      add   a,4E
D5E4  6F        ld    l,a         plus AD4E
D5E5  CEAD      adc   a,AD        gibt ADD0 - AE02 für 'A' - 'Z'
D5E7  95        sub   l
D5E8  67        ld    h,a
D5E9  C9        ret
```

```
*************************************** Tabellenadresse für Arrays berechnen
D5EA  2A87AE     ld     hl,(AE87)     Arraystart
D5ED  2B         dec    hl            minus 1
D5EE  44         ld     b,h
D5EF  4D         ld     c,l           nach bc
D5F0  E603       and    03
D5F2  3D         dec    a
D5F3  87         add    a,a
D5F4  C606       add    a,06
D5F6  6F         ld     l,a           plus AE06
D5F7  CEAE       adc    a,AE
D5F9  95         sub    l
D5FA  67         ld     h,a
D5FB  C9         ret

******************************************** Alle Variablen auf Typ REAL
D5FC  015A41     ld     bc,415A       'AZ'
D5FF  1E05       ld     e,05          'Real'
D601  79         ld     a,c
D602  90         sub    b             Anzahl nach a
D603  383D       jr     c,D642        kleiner 1, 'Syntax error'
D605  E5         push   hl
D606  3C         inc    a
D607  21CBAD     ld     hl,ADCB       Basis der Tabelle = ADCB+'A'
D60A  0600       ld     b,00
D60C  09         add    hl,bc         Buchstabe gleich Zeiger in Tabelle
D60D  73         ld     (hl),e        Typ abspeichern
D60E  2B         dec    hl
D60F  3D         dec    a             alle Buchstaben
D610  20FB       jr     nz,D60D
D612  E1         pop    hl
D613  C9         ret

************************************************ BASIC-Befehl DEFSTR
D614  1E03       ld     e,03          'String'
D616  1806       jr     D61E

************************************************* BASIC-Befehl DEFINT
D618  1E02       ld     e,02          'Integer'
D61A  1802       jr     D61E

************************************************ BASIC-Befehl DEFREAL
D61C  1E05       ld     e,05          'Real'
D61E  7E         ld     a,(hl)        Buchstabe holen
D61F  CD71FF     call   FF71          Test auf Buchstabe
D622  301E       jr     nc,D642       'Syntax error'
D624  4F         ld     c,a           nach bc (von - bis)
D625  47         ld     b,a
D626  CD3FDD     call   DD3F          Blanks überlesen
D629  FE2D       cp     2D            '-'
D62B  200C       jr     nz,D639
D62D  CD3FDD     call   DD3F          Blanks überlesen
D630  CD71FF     call   FF71          Test auf Buchstabe
D633  300D       jr     nc,D642       'Syntax error'
D635  4F         ld     c,a           bis
```

```
D636  CD3FDD        call   DD3F         Blanks überlesen
D639  CD01D6        call   D601         Variablentyp setzen
D63C  CD55DD        call   DD55         folgt Komma ?
D63F  38DD          jr     c,D61E       ja, nächste Variable verarbeiten
D641  C9            ret

D642  1E02          ld     e,02         'Syntax error'
D644  1806          jr     D64C

D646  1E09          ld     e,09         'Subscript out of range'
D648  1802          jr     D64C

D64A  1E0A          ld     e,0A         'Array already dimensioned'
D64C  C394CA        jp     CA94         Fehlermeldung ausgeben

*******************************************************
D64F  FEF8          cp     F8
D651  CAA0F1        jp     z,F1A0       Befehlserweiterung

**************************************************** BASIC-Befehl LET
D654  CD86D6        call   D686         Variable holen
D657  D5            push   de
D658  CD37DD        call   DD37         Test auf nachfolgendes Zeichen
D65B  EF            db     EF           '='
D65C  CDFBCE        call   CEFB         Ausdruck holen
D65F  78            ld     a,b
D660  E3            ex     (sp),hl
D661  CD66D6        call   D666         Wert an Variable zuweisen
D664  E1            pop    hl
D665  C9            ret

************************************************ Wert an Variable zuweisen
D666  47            ld     b,a          Variablentyp
D667  CD23FF        call   FF23         und Ergebnistyp
D66A  B8            cp     b            vergleichen
D66B  78            ld     a,b
D66C  C4D7FE        call   nz,FED7      Typanpassung, sonst 'Type mismatch'
D66F  CD45FF        call   FF45         Test auf String
D672  C262FF        jp     nz,FF62      nein, Variable nach (hl) kopieren
D675  E5            push   hl
D676  CD59FB        call   FB59         Stringverwaltung
D679  D1            pop    de
D67A  C366FF        jp     FF66         Zeiger auf String übernehmen

**************************************************** BASIC-Befehl DIM
D67D  CDB5D7        call   D7B5         Dimensionierung
D680  CD55DD        call   DD55         folgt Komma ?
D683  38F8          jr     c,D67D       ja, nächste Variable
D685  C9            ret

******************************************************* Variable suchen
D686  CD06D9        call   D906         Variablennamen lesen
D689  CDDBD7        call   D7DB         Test auf dimensionierte Variable
D68C  3842          jr     c,D6D0       Variablentyp holen
D68E  1828          jr     D6B8
```

```
********************************************* Variablenadresse holen
D690  CD06D9     call  D906      Variablennamen lesen
D693  CDDBD7     call  D7DB      Test auf dimensionierte Variable
D696  3838       jr    c,D6D0    Variablentyp holen
D698  E5         push  hl
D699  79         ld    a,c       erster Buchstabe
D69A  CDDBD5     call  D5DB      Tabellenposition berechnen
D69D  CDDED6     call  D6DE
D6A0, 182D       jr    D6CF

************************************************ Funktion suchen
D6A2  CD06D9     call  D906      Variablennamen lesen
D6A5  3821       jr    c,D6C8
D6A7  E5         push  hl
D6A8  CDD9D5     call  D5D9      Tabellenposition für FN berechnen
D6AB  CDDED6     call  D6DE
D6AE  D43DD7     call  nc,D73D   Funktion anlegen
D6B1  181C       jr    D6CF

**********************************************************
D6B3  CD06D9     call  D906      Variablennamen lesen
D6B6  3810       jr    c,D6C8
D6B8  E5         push  hl
D6B9  79         ld    a,c       erster Buchstabe
D6BA  CDDBD5     call  D5DB      Tabellenposition berechnen
D6BD  CDDED6     call  D6DE
D6C0  3AC1B0     ld    a,(B0C1)  Variablentyp
D6C3  D449D7     call  nc,D749
D6C6  1807       jr    D6CF

D6C8  E5         push  hl
D6C9  2A85AE     ld    hl,(AE85) Variablenstart
D6CC  2B         dec   hl
D6CD  19         add   hl,de
D6CE  EB         ex    de,hl
D6CF  E1         pop   hl
D6D0  3AC1B0     ld    a,(B0C1)  Variablentyp
D6D3  47         ld    b,a
D6D4  4F         ld    c,a
D6D5  C9         ret

**********************************************************
D6D6  CD06D9     call  D906      Variablennamen lesen
D6D9  CDC1E8     call  E8C1      Test auf indizierte Variable
D6DC  18F2       jr    D6D0      Variablentyp holen

**********************************************************
D6DE  D5         push  de
D6DF  EB         ex    de,hl
D6E0  2A2BAE     ld    hl,(AE2B)
D6E3  7C         ld    a,h
D6E4  B5         or    l
D6E5  280E       jr    z,D6F5
D6E7  D5         push  de
D6E8  23         inc   hl
```

```
D6E9  23            inc   hl
D6EA  C5            push  bc
D6EB  010000        ld    bc,0000
D6EE  CD08D7        call  D708         Array suchen
D6F1  C1            pop   bc
D6F2  3810          jr    c,D704       gefunden ?
D6F4  D1            pop   de
D6F5  EB            ex    de,hl
D6F6  E5            push  hl
D6F7  CD08D7        call  D708         Array suchen
D6FA  3803          jr    c,D6FF       gefunden ?
D6FC  E1            pop   hl
D6FD  D1            pop   de
D6FE  C9            ret

D6FF  F1            pop   af
D700  E1            pop   hl
D701  C36DD7        jp    D76D

D704  F1            pop   af
D705  F1            pop   af
D706  37            scf
D707  C9            ret

************************************************** Array suchen
D708  7E            ld    a,(hl)
D709  23            inc   hl
D70A  66            ld    h,(hl)
D70B  6F            ld    l,a
D70C  B4            or    h
D70D  C8            ret   z
D70E  09            add   hl,bc
D70F  E5            push  hl
D710  23            inc   hl
D711  23            inc   hl
D712  EB            ex    de,hl
D713  2A27AE        ld    hl,(AE27)
D716  1A            ld    a,(de)
D717  BE            cp    (hl)
D718  2014          jr    nz,D72E
D71A  23            inc   hl
D71B  13            inc   de
D71C  17            rla
D71D  30F7          jr    nc,D716
D71F  EB            ex    de,hl
D720  3AC1B0        ld    a,(B0C1)     Variablentyp
D723  3D            dec   a
D724  AE            xor   (hl)
D725  E607          and   07
D727  2005          jr    nz,D72E
D729  EB            ex    de,hl
D72A  13            inc   de
D72B  E1            pop   hl
D72C  37            scf
D72D  C9            ret
```

```
D72E  E1          pop   hl
D72F  18D7        jr    D708          Array suchen

*************************************************************
D731  F5          push  af
D732  54          ld    d,h
D733  5D          ld    e,l
D734  23          inc   hl
D735  23          inc   hl
D736  7E          ld    a,(hl)
D737  23          inc   hl
D738  17          rla
D739  30FB        jr    nc,D736
D73B  F1          pop   af
D73C  C9          ret

*************************************************************
D73D  3E02        ld    a,02
D73F  CD49D7      call  D749
D742  1B          dec   de
D743  1A          ld    a,(de)
D744  F640        or    40            Bit 6 setzen, 'FN'
D746  12          ld    (de),a
D747  13          inc   de
D748  C9          ret

*************************************************************
D749  D5          push  de
D74A  E5          push  hl
D74B  C5          push  bc
D74C  F5          push  af
D74D  CD77D7      call  D777
D750  F5          push  af
D751  2A87AE      ld    hl,(AE87)     Arraystart
D754  EB          ex    de,hl
D755  CDF8F5      call  F5F8          Platz im Variablenbereich reservieren
D758  CD3AF5      call  F53A          Zeiger für Arraybereich erhöhen
D75B  F1          pop   af
D75C  CD8AD7      call  D78A
D75F  F1          pop   af
D760  2B          dec   hl
D761  3600        ld    (hl),00
D763  3D          dec   a
D764  20FA        jr    nz,D760
D766  C1          pop   bc
D767  E3          ex    (sp),hl
D768  CDA5D7      call  D7A5
D76B  D1          pop   de
D76C  E1          pop   hl
D76D  23          inc   hl
D76E  7B          ld    a,e
D76F  91          sub   c
D770  77          ld    (hl),a
D771  23          inc   hl
D772  7A          ld    a,d
```

```
D773  98            sbc   a,b
D774  77            ld    (hl),a
D775  37            scf
D776  C9            ret

************************************************************
D777  C603          add   a,03
D779  4F            ld    c,a
D77A  2A27AE        ld    hl,(AE27)
D77D  0600          ld    b,00
D77F  0C            inc   c
D780  04            inc   b
D781  7E            ld    a,(hl)
D782  23            inc   hl
D783  17            rla
D784  30F9          jr    nc,D77F
D786  78            ld    a,b
D787  0600          ld    b,00
D789  C9            ret

************************************************************
D78A  62            ld    h,d
D78B  6B            ld    l,e
D78C  09            add   hl,bc
D78D  4F            ld    c,a
D78E  0600          ld    b,00
D790  E5            push  hl
D791  D5            push  de
D792  13            inc   de
D793  13            inc   de
D794  2A27AE        ld    hl,(AE27)
D797  CDF2FF        call  FFF2          ldir
D79A  3AC1B0        ld    a,(B0C1)      Variablentyp
D79D  3D            dec   a
D79E  12            ld    (de),a
D79F  13            inc   de
D7A0  42            ld    b,d
D7A1  4B            ld    c,e
D7A2  D1            pop   de
D7A3  E1            pop   hl
D7A4  C9            ret

************************************************************
D7A5  7E            ld    a,(hl)
D7A6  12            ld    (de),a
D7A7  7B            ld    a,e
D7A8  91            sub   c
D7A9  77            ld    (hl),a
D7AA  23            inc   hl
D7AB  7E            ld    a,(hl)
D7AC  F5            push  af
D7AD  7A            ld    a,d
D7AE  98            sbc   a,b
D7AF  77            ld    (hl),a
D7B0  F1            pop   af
```

```
D7B1  13                inc     de
D7B2  12                ld      (de),a
D7B3  13                inc     de
D7B4  C9                ret

******************************************** Dimensionierung
D7B5  CD06D9            call    D906          Variablenname holen
D7B8  7E                ld      a,(hl)
D7B9  FE28              cp      28            '('
D7BB  2805              jr      z,D7C2
D7BD  EE5B              xor     5B            '['
D7BF  C242D6            jp      nz,D642       'Syntax error'
D7C2  CD5AD8            call    D85A
D7C5  E5                push    hl
D7C6  C5                push    bc
D7C7  3AC1B0            ld      a,(B0C1)      Variablentyp
D7CA  CDEAD5            call    D5EA          Tabellenposition für Array berechnen
D7CD  CD08D7            call    D708          Array suchen
D7D0  DA4AD6            jp      c,D64A        gefunden, 'Array already dimensioned'
D7D3  C1                pop     bc
D7D4  3EFF              ld      a,FF
D7D6  CD8AD8            call    D88A
D7D9  E1                pop     hl
D7DA  C9                ret

**************************************** Test auf dimensionierte Variable
D7DB  F5                push    af
D7DC  7E                ld      a,(hl)
D7DD  FE28              cp      28            '('
D7DF  2810              jr      z,D7F1
D7E1  EE5B              xor     5B            '['
D7E3  280C              jr      z,D7F1
D7E5  F1                pop     af
D7E6  D0                ret     nc
D7E7  E5                push    hl
D7E8  2A85AE            ld      hl,(AE85)     Variablenstart
D7EB  2B                dec     hl
D7EC  19                add     hl,de
D7ED  EB                ex      de,hl
D7EE  E1                pop     hl
D7EF  37                scf
D7F0  C9                ret

******************************************* dimensionierte Variable
D7F1  CD5AD8            call    D85A
D7F4  F1                pop     af
D7F5  E5                push    hl
D7F6  3007              jr      nc,D7FF
D7F8  2A87AE            ld      hl,(AE87)     Arraystart
D7FB  2B                dec     hl
D7FC  19                add     hl,de
D7FD  1815              jr      D814
```

```
D7FF  C5       push  bc
D800  D5       push  de
D801  3AC1B0   ld    a,(B0C1)     Variablentyp
D804  CDEAD5   call  D5EA         Tabellenposition für Array berechnen
D807  CD08D7   call  D708         Array suchen
D80A  300F     jr    nc,D81B      nicht gefunden ?
D80C  13       inc   de
D80D  13       inc   de
D80E  E1       pop   hl
D80F  CD6DD7   call  D76D
D812  C1       pop   bc
D813  EB       ex    de,hl
D814  78       ld    a,b
D815  96       sub   (hl)
D816  C246D6   jp    nz,D646      'Subscript out of range'
D819  180A     jr    D825

D81B  E1       pop   hl
D81C  C1       pop   bc
D81D  AF       xor   a
D81E  CD8AD8   call  D88A
D821  CD6DD7   call  D76D
D824  EB       ex    de,hl
D825  110000   ld    de,0000
D828  46       ld    b,(hl)       Anzahl der Dimensionen
D829  23       inc   hl
D82A  E5       push  hl
D82B  D5       push  de
D82C  5E       ld    e,(hl)
D82D  23       inc   hl           Arraygrenze nach de
D82E  56       ld    d,(hl)
D82F  3E02     ld    a,02
D831  CDA0F5   call  F5A0         Platz im BASIC-Stack freigeben
D834  7E       ld    a,(hl)
D835  23       inc   hl           Index nach hl holen
D836  66       ld    h,(hl)
D837  6F       ld    l,a
D838  CDB8FF   call  FFB8         Vergleich hl <> de
D83B  D246D6   jp    nc,D646      'Subscript out of range'
D83E  E3       ex    (sp),hl
D83F  CDBEBD   call  BDBE         Integer-Multiplikation ohne Vorzeichen
D842  D1       pop   de
D843  19       add   hl,de
D844  EB       ex    de,hl
D845  E1       pop   hl
D846  23       inc   hl
D847  23       inc   hl
D848  05       dec   b            nächster Index
D849  20DF     jr    nz,D82A
D84B  E5       push  hl
D84C  2AC1B0   ld    hl,(B0C1)    Variablentyp
D84F  2600     ld    h,00
D851  CDBEBD   call  BDBE         Integer-Multiplikation ohne Vorzeichen
D854  D1       pop   de
D855  19       add   hl,de
```

```
D856  EB          ex    de,hl
D857  E1          pop   hl
D858  37          scf
D859  C9          ret

************************************************ Indices lesen
D85A  D5          push  de
D85B  CD3FDD      call  DD3F        Blanks überlesen
D85E  3AC1B0      ld    a,(B0C1)    Variablentyp
D861  F5          push  af
D862  0600        ld    b,00
D864  CD7CCE      call  CE7C        16-Bit-Wert 0 - 32767 holen, Index
D867  E5          push  hl
D868  3E02        ld    a,02
D86A  CDB0F5      call  F5B0        Platz im BASIC-Stack reservieren
D86D  73          ld    (hl),e
D86E  23          inc   hl          Index auf BASIC-Stack
D86F  72          ld    (hl),d
D870  E1          pop   hl
D871  04          inc   b
D872  CD55DD      call  DD55        folgt Komma ?
D875  38ED        jr    c,D864      ja, nächster Index
D877  7E          ld    a,(hl)
D878  FE29        cp    29          ')'
D87A  2805        jr    z,D881
D87C  FE5D        cp    5D          ']'
D87E  C242D6      jp    nz,D642     'Syntax error'
D881  CD3FDD      call  DD3F        Blanks überlesen
D884  F1          pop   af
D885  32C1B0      ld    (B0C1),a    Variablentyp
D888  D1          pop   de
D889  C9          ret

*****************************************************
D88A  E5          push  hl
D88B  3226AE      ld    (AE26),a
D88E  C5          push  bc
D88F  78          ld    a,b
D890  87          add   a,a
D891  C603        add   a,03
D893  CD77D7      call  D777
D896  F5          push  af
D897  2A89AE      ld    hl,(AE89)   Arrayende
D89A  EB          ex    de,hl
D89B  CDF8F5      call  F5F8        Platz im Variablenbereich reservieren
D89E  F1          pop   af
D89F  CD8AD7      call  D78A
D8A2  60          ld    h,b
D8A3  69          ld    l,c
D8A4  C1          pop   bc
D8A5  D5          push  de
D8A6  23          inc   hl
D8A7  23          inc   hl
D8A8  3AC1B0      ld    a,(B0C1)    Variablentyp
D8AB  5F          ld    e,a
```

```
D8AC 1600      ld    d,00
D8AE 70        ld    (hl),b
D8AF E5        push  hl
D8B0 23        inc   hl
D8B1 D5        push  de
D8B2 3A26AE    ld    a,(AE26)
D8B5 B7        or    a
D8B6 110B00    ld    de,000B       11, Defaultwert für Index
D8B9 280B      jr    z,D8C6
D8BB E5        push  hl
D8BC 3E02      ld    a,02
D8BE CDA0F5    call  F5A0          Platz im BASIC-Stack freigeben
D8C1 5E        ld    e,(hl)
D8C2 23        inc   hl
D8C3 56        ld    d,(hl)
D8C4 13        inc   de
D8C5 E1        pop   hl
D8C6 73        ld    (hl),e
D8C7 23        inc   hl
D8C8 72        ld    (hl),d
D8C9 23        inc   hl
D8CA E3        ex    (sp),hl
D8CB CDBEBD    call  BDBE          Integer-Multiplikation ohne Vorzeichen
D8CE DA46D6    jp    c,D646        'Subscript out of range'
D8D1 EB        ex    de,hl
D8D2 E1        pop   hl
D8D3 10DC      djnz  D8B1          nächster Index
D8D5 42        ld    b,d
D8D6 4B        ld    c,e
D8D7 54        ld    d,h
D8D8 5D        ld    e,l
D8D9 CDFBF5    call  F5FB          Speicherplatz reservieren
D8DC 2289AE    ld    (AE89),hl     Arrayende
D8DF C5        push  bc
D8E0 2B        dec   hl
D8E1 3600      ld    (hl),00
D8E3 0B        dec   bc
D8E4 78        ld    a,b
D8E5 B1        or    c
D8E6 20F8      jr    nz,D8E0
D8E8 C1        pop   bc
D8E9 E1        pop   hl
D8EA 5E        ld    e,(hl)
D8EB 1600      ld    d,00
D8ED EB        ex    de,hl
D8EE 29        add   hl,hl
D8EF 23        inc   hl
D8F0 09        add   hl,bc
D8F1 EB        ex    de,hl
D8F2 2B        dec   hl
D8F3 2B        dec   hl
D8F4 73        ld    (hl),e
D8F5 23        inc   hl
D8F6 72        ld    (hl),d
D8F7 23        inc   hl
```

```
D8F8  E3          ex    (sp),hl
D8F9  EB          ex    de,hl
D8FA  3AC1B0      ld    a,(B0C1)      Variablentyp
D8FD  CDEAD5      call  D5EA          Tabellenposition für Array berechnen
D900  CDA5D7      call  D7A5
D903  D1          pop   de
D904  E1          pop   hl
D905  C9          ret

******************************************** Variablenname holen
D906  CD7FD9      call  D97F          Variablentyp feststellen
D909  23          inc   hl
D90A  5E          ld    e,(hl)
D90B  23          inc   hl
D90C  56          ld    d,(hl)
D90D  7A          ld    a,d
D90E  B3          or    e             Variable schon angelegt ?
D90F  280A        jr    z,D91B        nein
D911  23          inc   hl
D912  7E          ld    a,(hl)        Buchstaben des Namens überlesen
D913  17          rla                 Bit 7 testen
D914  30FB        jr    nc,D911       letzter Buchstabe ?
D916  CD3FDD      call  DD3F          Blanks überlesen
D919  37          scf
D91A  C9          ret

*******************************************************************
D91B  2B          dec   hl
D91C  2B          dec   hl            Zeiger auf Variablentyp setzen
D91D  EB          ex    de,hl
D91E  C1          pop   bc
D91F  2A27AE      ld    hl,(AE27)
D922  E5          push  hl
D923  212BD9      ld    hl,D92B
D926  E5          push  hl
D927  C5          push  bc
D928  EB          ex    de,hl
D929  180E        jr    D939

D92B  E5          push  hl
D92C  2A27AE      ld    hl,(AE27)
D92F  CDACF5      call  F5AC          BASIC-Stackpointer setzen
D932  E1          pop   hl
D933  E3          ex    (sp),hl
D934  2227AE      ld    (AE27),hl
D937  E1          pop   hl
D938  C9          ret

D939  E5          push  hl
D93A  7E          ld    a,(hl)
D93B  23          inc   hl
D93C  23          inc   hl
D93D  23          inc   hl
D93E  4E          ld    c,(hl)
D93F  CBA9        res   5,c
```

```
D941  FE0B      cp    0B
D943  3819      jr    c,D95E
D945  79        ld    a,c
D946  E61F      and   1F
D948  C60B      add   a,0B
D94A  5F        ld    e,a          plus AE0B
D94B  CEAE      adc   a,AE
D94D  93        sub   e
D94E  57        ld    d,a
D94F  1A        ld    a,(de)
D950  32C1B0    ld    (B0C1),a     Variablentyp
D953  E3        ex    (sp),hl
D954  360D      ld    (hl),0D
D956  FE05      cp    05
D958  2803      jr    z,D95D
D95A  C609      add   a,09         05 + 09 => 0D
D95C  77        ld    (hl),a
D95D  E3        ex    (sp),hl
D95E  EB        ex    de,hl
D95F  3E28      ld    a,28         40
D961  CDB0F5    call  F5B0         Platz im BASIC-Stack reservieren
D964  2227AE    ld    (AE27),hl
D967  0629      ld    b,29         41
D969  05        dec   b            schon 40 Zeichen ?
D96A  CA42D6    jp    z,D642       ja, 'Syntax error'
D96D  1A        ld    a,(de)       nächstes Zeichen aus Namen holen
D96E  13        inc   de
D96F  E6DF      and   DF           Klein- in Großschreibung wandeln
D971  77        ld    (hl),a       abspeichern im BASIC-Stack
D972  23        inc   hl
D973  17        rla                letztes Zeichen ?
D974  30F3      jr    nc,D969      nein
D976  CDACF5    call  F5AC         BASIC-Stackpointer setzen
D979  EB        ex    de,hl
D97A  2B        dec   hl
D97B  D1        pop   de
D97C  C33FDD    jp    DD3F         Blanks überlesen

******************************************* Variablentyp feststellen
D97F  7E        ld    a,(hl)
D980  FE0B      cp    0B
D982  3802      jr    c,D986       kleiner 0B ?
D984  C6F7      add   a,F7         -9, 0D => 05
D986  FE04      cp    04           '!', Real-Variable ?
D988  2809      jr    z,D993       Typ auf 'real' setzen
D98A  3004      jr    nc,D990      'Syntax error'
D98C  FE02      cp    02           '%', Integer-Variable ?
D98E  3005      jr    nc,D995      oder '$', String ?
D990  C342D6    jp    D642         'Syntax error'

D993  3E05      ld    a,05         'Real'
D995  32C1B0    ld    (B0C1),a     Variablentyp merken
D998  C9        ret
```

```
*************************************************** Arraytabelle updaten
D999  CDC6D5     call   D5C6       Tabelle für Arrays löschen
D99C  2A89AE     ld     hl,(AE89)  Arrayende
D99F  EB         ex     de,hl
D9A0  2A87AE     ld     hl,(AE87)  Arraystart
D9A3  CDB8FF     call   FFB8       Vergleich hl <> de
D9A6  C8         ret    z
D9A7  D5         push   de
D9A8  CD31D7     call   D731
D9AB  7E         ld     a,(hl)
D9AC  23         inc    hl
D9AD  E607       and    07
D9AF  3C         inc    a
D9B0  E5         push   hl
D9B1  CDEAD5     call   D5EA       Tabellenposition für Array berechnen
D9B4  CDA5D7     call   D7A5
D9B7  E1         pop    hl
D9B8  5E         ld     e,(hl)
D9B9  23         inc    hl
D9BA  56         ld     d,(hl)
D9BB  23         inc    hl
D9BC  19         add    hl,de
D9BD  D1         pop    de
D9BE  18E3       jr     D9A3

*************************************************** BASIC-Befehl ERASE
D9C0  CD89E9     call   E989
D9C3  CDCCD9     call   D9CC       Array löschen
D9C6  CD55DD     call   DD55       folgt Komma ?
D9C9  38F8       jr     c,D9C3     ja, nächstes Array
D9CB  C9         ret

*************************************************** Array löschen
D9CC  CD06D9     call   D906       Variablenname holen
D9CF  E5         push   hl
D9D0  3AC1B0     ld     a,(B0C1)   Variablentyp
D9D3  CDEAD5     call   D5EA       Tabellenposition für Array berechnen
D9D6  CD08D7     call   D708       Array suchen
D9D9  E5         push   hl
D9DA  EB         ex     de,hl
D9DB  1E05       ld     e,05       'Improper argument'
D9DD  D294CA     jp     nc,CA94    nicht angelegt, Fehlermeldung ausgeben

D9E0  5E         ld     e,(hl)
D9E1  23         inc    hl         Adresse des Arrays nach de
D9E2  56         ld     d,(hl)
D9E3  23         inc    hl
D9E4  19         add    hl,de
D9E5  EB         ex     de,hl
D9E6  2A89AE     ld     hl,(AE89)  Arrayende
D9E9  CDCFFF     call   FFCF       hl := hl - de
D9EC  E3         ex     (sp),hl
D9ED  C1         pop    bc
D9EE  EB         ex     de,hl
D9EF  78         ld     a,b
```

```
D9F0  B1          or    c
D9F1  C4F2FF      call  nz,FFF2       ldir
D9F4  EB          ex    de,hl
D9F5  2289AE      ld    (AE89),hl     Arrayende
D9F8  CD99D9      call  D999          Arraytabelle updaten
D9FB  E1          pop   hl
D9FC  C9          ret

*******************************************************
D9FD  210000      ld    hl,0000
DA00  222BAE      ld    (AE2B),hl
DA03  2229AE      ld    (AE29),hl
DA06  C9          ret

*******************************************************
DA07  E5          push  hl
DA08  2A2BAE      ld    hl,(AE2B)
DA0B  E5          push  hl
DA0C  2A29AE      ld    hl,(AE29)
DA0F  EB          ex    de,hl
DA10  3E06        ld    a,06
DA12  CDB0F5      call  F5B0          Platz im BASIC-Stack reservieren
DA15  2229AE      ld    (AE29),hl
DA18  73          ld    (hl),e
DA19  23          inc   hl
DA1A  72          ld    (hl),d
DA1B  23          inc   hl
DA1C  AF          xor   a
DA1D  77          ld    (hl),a
DA1E  23          inc   hl
DA1F  77          ld    (hl),a
DA20  23          inc   hl
DA21  D1          pop   de
DA22  73          ld    (hl),e
DA23  23          inc   hl
DA24  72          ld    (hl),d
DA25  E1          pop   hl
DA26  C9          ret

*******************************************************
DA27  E5          push  hl
DA28  2A29AE      ld    hl,(AE29)
DA2B  222BAE      ld    (AE2B),hl
DA2E  E1          pop   hl
DA2F  C9          ret

*******************************************************
DA30  E5          push  hl
DA31  2A29AE      ld    hl,(AE29)
DA34  CDACF5      call  F5AC          BASIC-Stackpointer setzen
DA37  5E          ld    e,(hl)
DA38  23          inc   hl
DA39  56          ld    d,(hl)
DA3A  23          inc   hl
DA3B  EB          ex    de,hl
```

```
DA3C  2229AE        ld     (AE29),hl
DA3F  EB            ex     de,hl
DA40  23            inc    hl
DA41  23            inc    hl
DA42  5E            ld     e,(hl)
DA43  23            inc    hl
DA44  56            ld     d,(hl)
DA45  EB            ex     de,hl
DA46  222BAE        ld     (AE2B),hl
DA49  E1            pop    hl
DA4A  C9            ret

*******************************************************
DA4B  E5            push   hl
DA4C  3E02          ld     a,02
DA4E  CDB0F5        call   F5B0           Platz im BASIC-Stack reservieren
DA51  E3            ex     (sp),hl
DA52  CD7FD9        call   D97F           Variablentyp feststellen
DA55  CD39D9        call   D939
DA58  E3            ex     (sp),hl
DA59  EB            ex     de,hl
DA5A  2A29AE        ld     hl,(AE29)
DA5D  23            inc    hl
DA5E  23            inc    hl
DA5F  010000        ld     bc,0000
DA62  CDA5D7        call   D7A5
DA65  3AC1B0        ld     a,(B0C1)       Variablentyp
DA68  47            ld     b,a
DA69  3C            inc    a
DA6A  CDB0F5        call   F5B0           Platz im BASIC-Stack reservieren
DA6D  78            ld     a,b
DA6E  3D            dec    a
DA6F  77            ld     (hl),a
DA70  23            inc    hl
DA71  EB            ex     de,hl
DA72  E1            pop    hl
DA73  C9            ret

*******************************************************
DA74  2A29AE        ld     hl,(AE29)
DA77  7C            ld     a,h
DA78  B5            or     l
DA79  280E          jr     z,DA89
DA7B  4E            ld     c,(hl)
DA7C  23            inc    hl
DA7D  46            ld     b,(hl)
DA7E  23            inc    hl
DA7F  C5            push   bc
DA80  010000        ld     bc,0000
DA83  CDCEDA        call   DACE
DA86  E1            pop    hl
DA87  18EE          jr     DA77

*******************************************************
DA89  01411A        ld     bc,1A41        26 Buchstaben, 'A'
```

```
DA8C C5         push  bc
DA8D 79         ld    a,c          erster Buchstabe des Namens
DA8E CDDBD5     call  D5DB         Tabellenposition berechnen
DA91 CDCEDA     call  DACE
DA94 C1         pop   bc
DA95 0C         inc   c            nächster Buchstabe
DA96 05         dec   b            schon alle Buchstaben ?
DA97 20F3       jr    nz,DA8C
DA99 3E03       ld    a,03
DA9B CDEAD5     call  D5EA         Tabellenposition für Array berechnen
DA9E 4E         ld    c,(hl)
DA9F 23         inc   hl
DAA0 46         ld    b,(hl)
DAA1 78         ld    a,b
DAA2 B1         or    c
DAA3 C8         ret   z
DAA4 2A87AE     ld    hl,(AE87)    Arraystart
DAA7 2B         dec   hl
DAA8 09         add   hl,bc
DAA9 E5         push  hl
DAAA D5         push  de
DAAB CD31D7     call  D731
DAAE D1         pop   de
DAAF 23         inc   hl
DAB0 4E         ld    c,(hl)
DAB1 23         inc   hl
DAB2 46         ld    b,(hl)
DAB3 23         inc   hl
DAB4 E5         push  hl
DAB5 09         add   hl,bc
DAB6 E3         ex    (sp),hl
DAB7 4E         ld    c,(hl)
DAB8 23         inc   hl
DAB9 0600       ld    b,00
DABB 09         add   hl,bc
DABC 09         add   hl,bc
DABD C1         pop   bc
DABE CDBEFF     call  FFBE         Vergleich hl <> bc
DAC1 2808       jr    z,DACB
DAC3 CDE7DA     call  DAE7
DAC6 23         inc   hl
DAC7 23         inc   hl
DAC8 23         inc   hl
DAC9 18F3       jr    DABE

DACB E1         pop   hl
DACC 18D0       jr    DA9E

DACE 7E         ld    a,(hl)
DACF 23         inc   hl
DAD0 66         ld    h,(hl)
DAD1 6F         ld    l,a
DAD2 B4         or    h
DAD3 C8         ret   z
DAD4 09         add   hl,bc
```

```
DAD5 E5         push  hl
DAD6 D5         push  de
DAD7 CD31D7     call  D731
DADA D1         pop   de
DADB 7E         ld    a,(hl)
DADC 23         inc   hl
DADD E607       and   07
DADF FE02       cp    02
DAE1 CCE7DA     call  z,DAE7
DAE4 E1         pop   hl
DAE5 18E7       jr    DACE

DAE7 C5         push  bc
DAE8 D5         push  de
DAE9 E5         push  hl
DAEA 7E         ld    a,(hl)
DAEB 23         inc   hl
DAEC 4E         ld    c,(hl)
DAED 23         inc   hl
DAEE 46         ld    b,(hl)
DAEF EB         ex    de,hl
DAF0 B7         or    a
DAF1 C4F8FF     call  nz,FFF8     jp (hl)
DAF4 E1         pop   hl
DAF5 D1         pop   de
DAF6 C1         pop   bc
DAF7 C9         ret

****************************************** BASIC-Befehl LINE
DAF8 CD37DD     call  DD37        Test auf nachfolgendes Zeichen
DAFB A3         db    A3          'INPUT'
DAFC CDCBC1     call  C1CB        Kanalnummer holen
DAFF F5         push  af
DB00 CD89DB     call  DB89        evtl. Dialogstring ausgeben
DB03 CD86D6     call  D686        Variable suchen
DB06 CD3CFF     call  FF3C        Typ 'String', sonst 'Type mismatch'
DB09 E5         push  hl
DB0A D5         push  de
DB0B CD1ADB     call  DB1A        Eingabe von aktivem Gerät holen
DB0E CDDCF7     call  F7DC        String in Descriptorstack eintragen
DB11 E1         pop   hl
DB12 CD6FD6     call  D66F        Ergebnis an Variable zuweisen
DB15 E1         pop   hl
DB16 F1         pop   af
DB17 C3AFC1     jp    C1AF        Kanalnummer zurücksetzen

************************************** Eingabe von aktivem Gerät holen
DB1A CDC0C1     call  C1C0
DB1D D266DC     jp    nc,DC66     Eingabe von Kassette holen
DB20 CDA2C1     call  C1A2
DB23 F5         push  af
DB24 CDADDB     call  DBAD        Eingabe von Tastatur holen
DB27 F1         pop   af
DB28 C3A2C1     jp    C1A2
```

```
************************************************ BASIC-Befehl INPUT
DB2B  CDCBC1      call  C1CB        Kanalnummer holen
DB2E  F5          push  af
DB2F  CD47DB      call  DB47        Eingabe holen und umwandeln
DB32  D5          push  de
DB33  CD86D6      call  D686        Variable suchen
DB36  E3          ex    (sp),hl
DB37  3E00        ld    a,00
DB39  CDBCDB      call  DBBC
DB3C  E3          ex    (sp),hl
DB3D  CD55DD      call  DD55        auf Komma prüfen
DB40  38F1        jr    c,DB33
DB42  D1          pop   de
DB43  F1          pop   af
DB44  C3AFC1      jp    C1AF

***************************************** Eingabe holen und umwandeln
DB47  CDC0C1      call  C1C0
DB4A  303D        jr    nc,DB89     evtl. Dialogstring ausgeben
DB4C  CDA2C1      call  C1A2
DB4F  F5          push  af
DB50  E5          push  hl
DB51  CD89DB      call  DB89        evtl. Dialogstring ausgeben
DB54  3E3F        ld    a,3F        '?'
DB56  D456C3      call  nc,C356
DB59  3E20        ld    a,20        ' '
DB5B  D456C3      call  nc,C356
DB5E  E5          push  hl
DB5F  CDADDB      call  DBAD        Eingabe von Tastatur holen
DB62  EB          ex    de,hl
DB63  E1          pop   hl
DB64  CDD3DB      call  DBD3
DB67  3809        jr    c,DB72
DB69  2177DB      ld    hl,DB77     '?Redo from start'
DB6C  CD41C3      call  C341        ausgeben
DB6F  E1          pop   hl
DB70  18DE        jr    DB50

DB72  F1          pop   af
DB73  F1          pop   af
DB74  C3A2C1      jp    C1A2

*******************************************************
DB77  3F 52 65 64 6F 20 66 72       '?Redo from start'
DB7F  6F 6D 20 73 74 61 72 74
DB87  0A 00

****************************************** evtl. Dialogstring ausgeben
DB89  7E          ld    a,(hl)
DB8A  FE3B        cp    3B          ';'
DB8C  322DAE      ld    (AE2D),a    Trennzeichen merken
DB8F  CC3FDD      call  z,DD3F      Blanks überlesen
DB92  EE22        xor   22          '"'
DB94  C0          ret   nz
DB95  CDCBF7      call  F7CB        Dialogstring lesen
```

```
DB98  CDC0C1        call   C1C0
DB9B  F5            push   af
DB9C  DC28F8        call   c,F828        String ausgeben
DB9F  F1            pop    af
DBA0  D4DAFB        call   nc,FBDA       Stringparameter holen
DBA3  CD55DD        call   DD55          folgt Komma ?
DBA6  D8            ret    c             ja
DBA7  CD37DD        call   DD37          Test auf nachfolgendes Zeichen
DBAA  3B            db     3B            ';'
DBAB  B7            or     a
DBAC  C9            ret

*************************************** Eingabe von Tastatur holen
DBAD  CD3BCA        call   CA3B          Eingabezeile holen
DBB0  D26BCB        jp     nc,CB6B       ESC gedrückt ?
DBB3  3A2DAE        ld     a,(AE2D)      Trennzeichen
DBB6  FE3B          cp     3B            ';'
DBB8  C44EC3        call   nz,C34E       kein ';', neue Zeile (LF ausgeben)
DBBB  C9            ret

**********************************************************
DBBC  D5            push   de
DBBD  CD02DC        call   DC02
DBC0  300C          jr     nc,DBCE
DBC2  E3            ex     (sp),hl
DBC3  CD66D6        call   D666          Wert an Variable zuweisen
DBC6  E1            pop    hl
DBC7  7E            ld     a,(hl)
DBC8  23            inc    hl
DBC9  B7            or     a
DBCA  C8            ret    z
DBCB  EE2C          xor    2C            ','
DBCD  C9            ret

DBCE  1E0D          ld     e,0D          'Type mismatch'
DBD0  C394CA        jp     CA94          Fehlermeldung ausgeben

**********************************************************
DBD3  D5            push   de
DBD4  E5            push   hl
DBD5  D5            push   de
DBD6  CDD6D6        call   D6D6          Variablenname und -typ holen
DBD9  E3            ex     (sp),hl
DBDA  AF            xor    a
DBDB  CD02DC        call   DC02
DBDE  301E          jr     nc,DBFE
DBE0  FE03          cp     03            'String'
DBE2  CCDAFB        call   z,FBDA        ja, Stringparameter holen
DBE5  E3            ex     (sp),hl
DBE6  CD55DD        call   DD55          folgt Komma ?
DBE9  E3            ex     (sp),hl
DBEA  300B          jr     nc,DBF7       nein
DBEC  CD61DD        call   DD61          Blank, TAB und LF überlesen
DBEF  EE2C          xor    2C            ','
DBF1  200B          jr     nz,DBFE
```

```
DBF3  23        inc   hl
DBF4  E3        ex    (sp),hl
DBF5  18DF      jr    DBD6

DBF7  CD61DD    call  DD61        Blank, TAB und LF überlesen
DBFA  B7        or    a
DBFB  2001      jr    nz,DBFE
DBFD  37        scf
DBFE  E1        pop   hl
DBFF  E1        pop   hl
DC00  D1        pop   de
DC01  C9        ret

DC02  5F        ld    e,a
DC03  CD45FF    call  FF45        Test auf String
DC06  57        ld    d,a
DC07  D5        push  de
DC08  2006      jr    nz,DC10
DC0A  CD21DC    call  DC21
DC0D  37        scf
DC0E  1809      jr    DC19

DC10  CDC0C1    call  C1C0
DC13  D438DC    call  nc,DC38
DC16  CDA3EC    call  ECA3
DC19  F5        push  af
DC1A  DC61DD    call  c,DD61      Blank, TAB und LF überlesen
DC1D  F1        pop   af
DC1E  D1        pop   de
DC1F  7A        ld    a,d
DC20  C9        ret

DC21  CDC0C1    call  C1C0
DC24  3806      jr    c,DC2C
DC26  CD47DC    call  DC47
DC29  C3DCF7    jp    F7DC        String in Descriptorstack eintragen

DC2C  CD61DD    call  DD61        Blank, TAB und LF überlesen
DC2F  FE22      cp    22          '"'
DC31  CACBF7    jp    z,F7CB      String lesen
DC34  7B        ld    a,e
DC35  C3E6F7    jp    F7E6

DC38  CD9DDC    call  DC9D
DC3B  3005      jr    nc,DC42     'EOF met'
DC3D  11C6DC    ld    de,DCC6
DC40  182C      jr    DC6E

DC42  1E18      ld    e,18        'EOF met'
DC44  C394CA    jp    CA94        Fehlermeldung ausgeben

DC47  CD9DDC    call  DC9D
DC4A  30F6      jr    nc,DC42     'EOF met'
DC4C  FE22      cp    22          '"'
DC4E  2805      jr    z,DC55
```

```
DC50  11CADC      ld     de,DCCA
DC53  1819        jr     DC6E

DC55  CDA8DC      call   DCA8
DC58  1163DC      ld     de,DC63
DC5B  3811        jr     c,DC6E
DC5D  21A4AC      ld     hl,ACA4       Start des Eingabepuffers
DC60  3600        ld     (hl),00       erstes Zeichen gleich 00
DC62  C9          ret

DC63  FE22        cp     22            '"'
DC65  C9          ret

DC66  CDA8DC      call   DCA8
DC69  30D7        jr     nc,DC42       'EOF met'
DC6B  11CDDC      ld     de,DCCD
DC6E  21A4AC      ld     hl,ACA4       Start des Eingabepuffers
DC71  E5          push   hl
DC72  06FF        ld     b,FF
DC74  CDFBFF      call   FFFB          jp (de)
DC77  280C        jr     z,DC85
DC79  77          ld     (hl),a
DC7A  23          inc    hl
DC7B  05          dec    b
DC7C  2805        jr     z,DC83
DC7E  CDA8DC      call   DCA8
DC81  38F1        jr     c,DC74
DC83  F6FF        or     FF
DC85  3600        ld     (hl),00
DC87  E1          pop    hl
DC88  C0          ret    nz
DC89  FE0D        cp     0D            CR
DC8B  C8          ret    z
DC8C  FE22        cp     22            '"'
DC8E  C4D0DC      call   nz,DCD0
DC91  C0          ret    nz
DC92  CD9DDC      call   DC9D
DC95  D0          ret    nc
DC96  CDCADC      call   DCCA
DC99  C414C4      call   nz,C414       CAS RETURN
DC9C  C9          ret

DC9D  CDA8DC      call   DCA8
DCA0  D0          ret    nc
DCA1  CDD0DC      call   DCD0
DCA4  28F7        jr     z,DC9D
DCA6  37          scf
DCA7  C9          ret

DCA8  CD24C4      call   C424          ein Zeichen einlesen
DCAB  D0          ret    nc
DCAC  C5          push   bc
DCAD  FE0D        cp     0D            CR
DCAF  060A        ld     b,0A          LF
DCB1  2805        jr     z,DCB8
```

```
DCB3 B8         cp    b
DCB4 200D       jr    nz,DCC3
DCB6 060D       ld    b,0D          CR
DCB8 4F         ld    c,a
DCB9 CD24C4     call  C424          ein Zeichen einlesen
DCBC 3004       jr    nc,DCC2
DCBE B8         cp    b
DCBF C414C4     call  nz,C414       CAS RETURN
DCC2 79         ld    a,c
DCC3 C1         pop   bc
DCC4 37         scf
DCC5 C9         ret

DCC6 CDD0DC     call  DCD0
DCC9 C8         ret   z
DCCA FE2C       cp    2C            ','
DCCC C8         ret   z
DCCD FE0D       cp    0D            CR
DCCF C9         ret

DCD0 FE20       cp    20            ' '
DCD2 C8         ret   z
DCD3 FE09       cp    09            TAB
DCD5 C8         ret   z
DCD6 FE0A       cp    0A            LF
DCD8 C9         ret

****************************************** BASIC-Befehl RESTORE
DCD9 280A       jr    z,DCE5
DCDB CDE1CE     call  CEE1          Zeilennummer nach de holen
DCDE E5         push  hl
DCDF CD9AE7     call  E79A          BASIC-Zeile de suchen
DCE2 2B         dec   hl
DCE3 182D       jr    DD12          DATA-Zeiger setzen

DCE5 E5         push  hl
DCE6 2A81AE     ld    hl,(AE81)     Programmstart
DCE9 1827       jr    DD12          als DATA-Zeiger

********************************************* BASIC-Befehl READ
DCEB E5         push  hl
DCEC 2A30AE     ld    hl,(AE30)     DATA-Zeiger
DCEF CD17DD     call  DD17          nächstes DATA-Element holen
DCF2 E3         ex    (sp),hl
DCF3 CD86D6     call  D686          Variable suchen
DCF6 E3         ex    (sp),hl
DCF7 23         inc   hl
DCF8 3E3A       ld    a,3A          ':'
DCFA CDBCDB     call  DBBC
DCFD 2B         dec   hl
DCFE 280B       jr    z,DD0B
DD00 2A2EAE     ld    hl,(AE2E)     Zeilenadresse während READ-Befehl
DD03 CDCEDD     call  DDCE          aktuelle Zeilenadresse setzen
DD06 1E02       ld    e,02          'Syntax error'
DD08 C394CA     jp    CA94          Fehlermeldung ausgeben
```

```
DD0B E3        ex    (sp),hl
DD0C CD55DD    call  DD55          folgt Komma ?
DD0F E3        ex    (sp),hl
DD10 38DD      jr    c,DCEF        ja
DD12 2230AE    ld    (AE30),hl     DATA-Zeiger
DD15 E1        pop   hl
DD16 C9        ret

DD17 7E        ld    a,(hl)
DD18 FE2C      cp    2C            ','
DD1A C8        ret   z
DD1B CDEFE8    call  E8EF          Rest der Zeile überlesen
DD1E B7        or    a             Zeilenende ?
DD1F 200E      jr    nz,DD2F       nein
DD21 23        inc   hl
DD22 7E        ld    a,(hl)        Zeilenlänge
DD23 23        inc   hl
DD24 B6        or    (hl)          Null, Programmende ?
DD25 23        inc   hl
DD26 1E04      ld    e,04          'DATA exhausted'
DD28 CA94CA    jp    z,CA94        Fehlermeldung ausgeben

DD2B 222EAE    ld    (AE2E),hl     Zeilenadresse während READ-Befehl
DD2E 23        inc   hl
DD2F CD3FDD    call  DD3F          Blanks überlesen
DD32 FE8C      cp    8C            'DATA'
DD34 20E5      jr    nz,DD1B
DD36 C9        ret

****************************************** Test auf nachfolgendes Zeichen
DD37 E3        ex    (sp),hl       Zeiger hinter CALL-Befehl
DD38 7E        ld    a,(hl)        nachfolgendes Zeichen holen
DD39 23        inc   hl
DD3A E3        ex    (sp),hl       Rücksprungadresse erhöhen
DD3B BE        cp    (hl)          Zeichen mit Programmtext vergleichen
DD3C C2C6DD    jp    nz,DDC6       nicht gleich, 'Syntax error'

************************************************** Blanks überlesen
DD3F 23        inc   hl
DD40 7E        ld    a,(hl)
DD41 FE20      cp    20            ' '
DD43 28FA      jr    z,DD3F        Blanks überlesen
DD45 FE01      cp    01
DD47 D0        ret   nc
DD48 B7        or    a             Zeilenende, Z=1
DD49 C9        ret

************************************* Ende des Statements, sonst 'Syntax error'
DD4A 7E        ld    a,(hl)        laufendes Zeichen
DD4B FE02      cp    02            kleiner 2 ?
DD4D D8        ret   c             ok
DD4E C3C6DD    jp    DDC6          sonst 'Syntax error'
```

```
****************************************** Test auf Ende des Statements
DD51  7E        ld    a,(hl)       laufenden Zeichen
DD52  FE02      cp    02           kleiner 2 ?
DD54  C9        ret

************************************** nächstes Zeichen auf Komma prüfen
DD55  2B        dec   hl
DD56  CD3FDD    call  DD3F         Blanks überlesen
DD59  EE2C      xor   2C           ','
DD5B  C0        ret   nz           nicht gefunden, c=0
DD5C  CD3FDD    call  DD3F         Blanks überlesen
DD5F  37        scf                gefunden, c=1
DD60  C9        ret

****************************************** Blank, TAB und LF überlesen
DD61  7E        ld    a,(hl)
DD62  23        inc   hl
DD63  FE20      cp    20           ' '
DD65  28FA      jr    z,DD61
DD67  FE09      cp    09           TAB
DD69  28F6      jr    z,DD61
DD6B  FE0A      cp    0A           LF
DD6D  28F2      jr    z,DD61
DD6F  2B        dec   hl
DD70  C9        ret

************************************************** Interpreterschleife
DD71  2A34AE    ld    hl,(AE34)    Adresse des aktuellen Statements
DD74  EB        ex    de,hl        Programmzeiger merken
DD75  2A8BB0    ld    hl,(B08B)    BASIC-Stackpointer
DD78  2232AE    ld    (AE32),hl    Speicher für BASIC-Stackpointer
DD7B  EB        ex    de,hl        Programmzeiger
DD7C  2234AE    ld    (AE34),hl    als Adresse des aktuellen Statements
DD7F  CD21B9    call  B921         KL POLL SYNCHRONOUS
DD82  DC07C8    call  c,C807       Event-Verarbeitung AFTER/EVERY
DD85  CD3FDD    call  DD3F         Blanks überlesen
DD88  C4ABDD    call  nz,DDAB      BASIC-Befehl ausführen
DD8B  7E        ld    a,(hl)       Programmtext lesen
DD8C  FE01      cp    01           ':', Ende des Statements ?
DD8E  28E4      jr    z,DD74       ja
DD90  3034      jr    nc,DDC6      'Syntax error'
DD92  23        inc   hl
DD93  7E        ld    a,(hl)
DD94  23        inc   hl           Zeilenlänge
DD95  B6        or    (hl)         gleich Null ?
DD96  23        inc   hl
DD97  280F      jr    z,DDA8       ja, zum END-Befehl
DD99  2236AE    ld    (AE36),hl    aktuelle Zeilenadresse merken
DD9C  23        inc   hl
DD9D  3A38AE    ld    a,(AE38)     TRACE-Flag gesetzt ?
DDA0  B7        or    a
DDA1  28D1      jr    z,DD74       nein
DDA3  CDEBDD    call  DDEB         TRACE-Routine
DDA6  18CC      jr    DD74         zur Interpreterschleife
```

```
DDA8 C376CB          jp    CB76          zum END-Befehl

********************************************** BASIC-Befehl ausführen
DDAB 87              add   a,a           Token mal 2
DDAC D24FD6          jp    nc,D64F       Test auf Befehlserweiterung
DDAF FEB9            cp    B9
DDB1 3010            jr    nc,DDC3       ungültiges Token, 'Syntax error'
DDB3 EB              ex    de,hl
DDB4 C601            add   a,01
DDB6 6F              ld    l,a           plus DE01 (Tabellenadresse)
DDB7 CEDE            adc   a,DE
DDB9 95              sub   l
DDBA 67              ld    h,a
DDBB 4E              ld    c,(hl)
DDBC 23              inc   hl
DDBD 46              ld    b,(hl)
DDBE C5              push  bc            Befehlsadresse auf Stack
DDBF EB              ex    de,hl
DDC0 C33FDD          jp    DD3F          Blanks überlesen, Sprung auf Befehl

****************************************************************
DDC3 CD07AC          call  AC07          ret
DDC6 1E02            ld    e,02          'Syntax error'
DDC8 C394CA          jp    CA94          Fehlermeldung ausgeben

***************************************** aktuelle Zeilenadresse auf Null
DDCB 210000          ld    hl,0000
DDCE 2236AE          ld    (AE36),hl     aktuelle Zeilenadresse
DDD1 C9              ret

****************************************** aktuelle Zeilenadresse laden
DDD2 2A36AE          ld    hl,(AE36)     aktuelle Zeilenadresse
DDD5 C9              ret

************************************ Test Direktmodus / Zeilennummer holen
DDD6 2A36AE          ld    hl,(AE36)     aktuelle Zeilenadresse
DDD9 7C              ld    a,h
DDDA B5              or    l
DDDB C8              ret   z             Null, Direktmodus
DDDC 7E              ld    a,(hl)
DDDD 23              inc   hl
DDDE 66              ld    h,(hl)        Zeilennummer nach hl
DDDF 6F              ld    l,a
DDE0 37              scf
DDE1 C9              ret

******************************************************* BASIC-Befehl TRON
DDE2 3EFF            ld    a,FF
DDE4 1801            jr    DDE7

****************************************************** BASIC-Befehl TROFF
DDE6 AF              xor   a
DDE7 3238AE          ld    (AE38),a      TRACE-Flag
DDEA C9              ret
```

```
************************************************** TRACE-Routine
DDEB 3E5B         ld    a,5B          '['
DDED CD56C3       call  C356          ausgeben
DDF0 E5           push  hl
DDF1 2A36AE       ld    hl,(AE36)     aktuelle Zeilenadresse
DDF4 7E           ld    a,(hl)
DDF5 23           inc   hl            Zeilennummer nach hl
DDF6 66           ld    h,(hl)
DDF7 6F           ld    l,a
DDF8 CD79EE       call  EE79          Zeilennummer ausgeben
DDFB E1           pop   hl
DDFC 3E5D         ld    a,5D          ']'
DDFE C356C3       jp    C356          ausgeben

****************************************** Adressen der BASIC-Befehle
DE01 71C9         dw    C971          80 AFTER
DE03 DFC0         dw    C0DF          81 AUTO
DE05 21C2         dw    C221          82 BORDER
DE07 BAF1         dw    F1BA          83 CALL
DE09 46D2         dw    D246          84 CAT
DE0B 3CEA         dw    EA3C          85 CHAIN
DE0D 32C1         dw    C132          86 CLEAR
DE0F B5C4         dw    C4B5          87 CLG
DE11 98D2         dw    D298          88 CLOSEIN
DE13 A1D2         dw    D2A1          89 CLOSEOUT
DE15 5AC2         dw    C25A          8A CLS
DE17 C0CB         dw    CBC0          8B CONT
DE19 EFE8         dw    E8EF          8C DATA
DE1B 17D1         dw    D117          8D DEF
DE1D 18D6         dw    D618          8E DEFINT
DE1F 1CD6         dw    D61C          8F DEFREAL
DE21 14D6         dw    D614          90 DEFSTR
DE23 E7D4         dw    D4E7          91 DEG
DE25 28E7         dw    E728          92 DELETE
DE27 7DD6         dw    D67D          93 DIM
DE29 C6C4         dw    C4C6          94 DRAW
DE2B CBC4         dw    C4CB          95 DRAWR
DE2D 52C0         dw    C052          96 EDIT
DE2F F3E8         dw    E8F3          97 ELSE
DE31 65CB         dw    CB65          98 END
DE33 85D3         dw    D385          99 ENT
DE35 4ED3         dw    D34E          9A ENV
DE37 C0D9         dw    D9C0          9B ERASE
DE39 8FCA         dw    CA8F          9C ERROR
DE3B 79C9         dw    C979          9D EVERY
DE3D 29C5         dw    C529          9E FOR
DE3F EDC6         dw    C6ED          9F GOSUB
DE41 E8C6         dw    C6E8          A0 GOTO
DE43 C7C6         dw    C6C7          A1 IF
DE45 2AC2         dw    C22A          A2 INK
DE47 2BDB         dw    DB2B          A3 INPUT
DE49 39D4         dw    D439          A4 KEY
DE4B 54D6         dw    D654          A5 LET
DE4D F8DA         dw    DAF8          A6 LINE
DE4F F7E0         dw    E0F7          A7 LIST
```

```
DE51  F6E9    dw   E9F6    A8 LOAD
DE53  D2C2    dw   C2D2    A9 LOCATE
DE55  EFF4    dw   F4EF    AA MEMORY
DE57  A6EA    dw   EAA6    AB MERGE
DE59  93F9    dw   F993    AC MID$
DE5B  4FC2    dw   C24F    AD MODE
DE5D  05C5    dw   C505    AE MOVE
DE5F  0AC5    dw   C50A    AF MOVER
DE61  FBC5    dw   C5FB    B0 NEXT
DE63  2BC1    dw   C12B    B1 NEW
DE65  E3C7    dw   C7E3    B2 ON
DE67  CBC8    dw   C8CB    B3 ON BREAK
DE69  F8CB    dw   CBF8    B4 ON ERROR GOTO 0
DE6B  40C9    dw   C940    B5 ON SQ
DE6D  5FD2    dw   D25F    B6 OPENIN
DE6F  56D2    dw   D256    B7 OPENOUT
DE71  8CC4    dw   C48C    B8 ORIGIN
DE73  77F1    dw   F177    B9 OUT
DE75  0AC2    dw   C20A    BA PAPER
DE77  12C2    dw   C212    BB PEN
DE79  D0C4    dw   C4D0    BC PLOT
DE7B  D5C4    dw   C4D5    BD PLOTR
DE7D  5FF1    dw   F15F    BE POKE
DE7F  FDF1    dw   F1FD    BF PRINT
DE81  F3E8    dw   E8F3    C0 '
DE83  EBD4    dw   D4EB    C1 RAD
DE85  59D5    dw   D559    C2 RANDOMIZE
DE87  EBDC    dw   DCEB    C3 READ
DE89  1ED3    dw   D31E    C4 RELEASE
DE8B  F3E8    dw   E8F3    C5 REM
DE8D  DFE7    dw   E7DF    C6 RENUM
DE8F  D9DC    dw   DCD9    C7 RESTORE
DE91  03CC    dw   CC03    C8 RESUME
DE93  0FC7    dw   C70F    C9 RETURN
DE95  BDE9    dw   E9BD    CA RUN
DE97  09EC    dw   EC09    CB SAVE
DE99  C0D2    dw   D2C0    CC SOUND
DE9B  94D4    dw   D494    CD SPEED
DE9D  5ACB    dw   CB5A    CE STOP
DE9F  9DF6    dw   F69D    CF SYMBOL
DEA1  19C3    dw   C319    D0 TAG
DEA3  20C3    dw   C320    D1 TAG OFF
DEA5  E6DD    dw   DDE6    D2 TRON
DEA7  E2DD    dw   DDE2    D3 TROFF
DEA9  7DF1    dw   F17D    D4 WAIT
DEAB  76C7    dw   C776    D5 WEND
DEAD  47C7    dw   C747    D6 WHILE
DEAF  E3C3    dw   C3E3    D7 WIDTH
DEB1  E1C2    dw   C2E1    D8 WINDOW
DEB3  7BF4    dw   F47B    D9 ZONE
DEB5  F6F1    dw   F1F6    DA WRITE
DEB7  E1C8    dw   C8E1    DB DI
DEB9  E7C8    dw   C8E7    DC EI
```

```
****************************************************************
DEBB D5         push  de
DEBC EB         ex    de,hl
DEBD 2A7FAE     ld    hl,(AE7F)     Beginn des freien RAMs
DEC0 EB         ex    de,hl
DEC1 D5         push  de
DEC2 AF         xor   a
DEC3 3239AE     ld    (AE39),a
DEC6 012C01     ld    bc,012C       max. 300 Zeichen
DEC9 CDE1DE     call  DEE1          Zeichen aus Eingabepuffer holen
DECC 7E         ld    a,(hl)
DECD B7         or    a             letztes Zeichen ?
DECE 20F9       jr    nz,DEC9       nein
DED0 3E2D       ld    a,2D
DED2 91         sub   c             301 - Zählerstand
DED3 4F         ld    c,a
DED4 3E01       ld    a,01          gleich Zeilenlänge
DED6 98         sbc   a,b
DED7 47         ld    b,a           nach b
DED8 AF         xor   a
DED9 12         ld    (de),a
DEDA 13         inc   de            dreimal Null als Abschluß
DEDB 12         ld    (de),a
DEDC 13         inc   de
DEDD 12         ld    (de),a
DEDE E1         pop   hl
DEDF D1         pop   de
DEE0 C9         ret

****************************************** Zeichen aus Eingabepuffer holen
DEE1 CD10AC     call  AC10          ret
DEE4 7E         ld    a,(hl)
DEE5 B7         or    a             letztes Zeichen ?
DEE6 C8         ret   z             ja
DEE7 CD71FF     call  FF71          Buchstabe ?
DEEA 381D       jr    c,DF09        ja
DEEC CD7FFF     call  FF7F          numerisch ?
DEEF DAFFDF     jp    c,DFFF        ja
DEF2 FE26       cp    26            '&' ?
DEF4 CA5AE0     jp    z,E05A        ja
DEF7 23         inc   hl
DEF8 FE80       cp    80            Token ?
DEFA D0         ret   nc            ja
DEFB FE20       cp    20            ' '
DEFD C280E0     jp    nz,E080
DF00 3A00AC     ld    a,(AC00)      zusätzliche Blanks ignorieren ?
DF03 B7         or    a
DF04 C0         ret   nz            ja
DF05 3E20       ld    a,20          ' '
DF07 181C       jr    DF25          in Puffer schreiben

DF09 CD4EDF     call  DF4E
DF0C D8         ret   c
DF0D FEC5       cp    C5            'REM'
DF0F CAEDE0     jp    z,E0ED
```

```
DF12  E5         push  hl
DF13  2130DF     ld    hl,DF30     Basisadresse der Tabelle
DF16  CDAAFF     call  FFAA        Tabelle durchsuchen
DF19  E1         pop   hl
DF1A  3819       jr    c,DF35      gefunden ? dann Rest nicht konvertieren
DF1C  F5         push  af
DF1D  FE97       cp    97          'ELSE'
DF1F  3E01       ld    a,01
DF21  CC25DF     call  z,DF25      in Puffer schreiben
DF24  F1         pop   af
DF25  12         ld    (de),a      Zeichen in Puffer schreiben
DF26  13         inc   de          Pufferzeiger erhöhen
DF27  0B         dec   bc          Zähler erniedrigen
DF28  79         ld    a,c
DF29  B0         or    b
DF2A  C0         ret   nz
DF2B  1E17       ld    e,17        'Line too long'
DF2D  C394CA     jp    CA94        Fehlermeldung ausgeben

*************************************************** spezielle Tokens
DF30  8C         db    8C'         DATA'
DF31  8E         db    8E'         DEFINT'
DF32  90         db    90'         DEFSTR'
DF33  8F         db    8F'         DEFREAL'
DF34  00         db    00          Tabellenende

*****************************************************************
DF35  CD25DF     call  DF25        in Puffer schreiben
DF38  7E         ld    a,(hl)
DF39  B7         or    a
DF3A  C8         ret   z
DF3B  FE3A       cp    3A          ':'
DF3D  280A       jr    z,DF49
DF3F  23         inc   hl
DF40  FE22       cp    22          '"'
DF42  20F1       jr    nz,DF35
DF44  CDBFE0     call  E0BF
DF47  18EF       jr    DF38

DF49  AF         xor   a
DF4A  3239AE     ld    (AE39),a
DF4D  C9         ret

DF4E  C5         push  bc
DF4F  D5         push  de
DF50  E5         push  hl
DF51  CD16AC     call  AC16        ret
DF54  7E         ld    a,(hl)
DF55  23         inc   hl
DF56  CD8AFF     call  FF8A        Klein- in Großbuchstaben wandeln
DF59  CDDDE2     call  E2DD        Adresse der Befehlsworte berechnen
DF5C  CD27E3     call  E327
DF5F  3028       jr    nc,DF89
DF61  79         ld    a,c
DF62  E67F       and   7F
```

```
DF64  CD7BFF   call   FF7B      Test auf Buchstabe oder Ziffer
DF67  300B     jr     nc,DF74
DF69  1A       ld     a,(de)
DF6A  FEE4     cp     E4        'FN'
DF6C  2806     jr     z,DF74
DF6E  7E       ld     a,(hl)
DF6F  CD7BFF   call   FF7B      Test auf Buchstabe oder Ziffer
DF72  3815     jr     c,DF89
DF74  F1       pop    af
DF75  1A       ld     a,(de)
DF76  B7       or     a
DF77  FAC8DF   jp     m,DFC8
DF7A  D1       pop    de
DF7B  C1       pop    bc
DF7C  F5       push   af
DF7D  3EFF     ld     a,FF      'Funktion'
DF7F  CD25DF   call   DF25      in Puffer schreiben
DF82  F1       pop    af
DF83  CD25DF   call   DF25      in Puffer schreiben
DF86  AF       xor    a
DF87  183A     jr     DFC3

DF89  E1       pop    hl
DF8A  D1       pop    de
DF8B  C1       pop    bc
DF8C  E5       push   hl
DF8D  2B       dec    hl
DF8E  23       inc    hl
DF8F  7E       ld     a,(hl)
DF90  CD7BFF   call   FF7B      Test auf Buchstabe oder Ziffer
DF93  38F9     jr     c,DF8E
DF95  CDEADF   call   DFEA
DF98  3804     jr     c,DF9E
DF9A  3E0D     ld     a,0D      Token für Variable
DF9C  1806     jr     DFA4

DF9E  23       inc    hl
DF9F  FE05     cp     05
DFA1  2001     jr     nz,DFA4
DFA3  3D       dec    a
DFA4  CD25DF   call   DF25      in Puffer schreiben
DFA7  AF       xor    a         Null
DFA8  CD25DF   call   DF25      in Puffer schreiben
DFAB  AF       xor    a
DFAC  CD25DF   call   DF25      in Puffer schreiben
DFAF  E3       ex     (sp),hl
DFB0  7E       ld     a,(hl)
DFB1  CD7BFF   call   FF7B      Test auf Buchstabe oder Ziffer
DFB4  3007     jr     nc,DFBD
DFB6  7E       ld     a,(hl)
DFB7  CD25DF   call   DF25      in Puffer schreiben
DFBA  23       inc    hl
DFBB  18F3     jr     DFB0
```

```
DFBD  CDDFE0      call   E0DF
DFC0  E1          pop    hl
DFC1  3EFF        ld     a,FF
DFC3  3239AE      ld     (AE39),a
DFC6  37          scf
DFC7  C9          ret

DFC8  E5          push   hl
DFC9  4F          ld     c,a
DFCA  21DCDF      ld     hl,DFDC      Basisadresse der Tabelle
DFCD  CDAAFF      call   FFAA         Tabelle durchsuchen
DFD0  9F          sbc    a,a
DFD1  E601        and    01
DFD3  3239AE      ld     (AE39),a
DFD6  79          ld     a,c
DFD7  E1          pop    hl
DFD8  D1          pop    de
DFD9  C1          pop    bc
DFDA  B7          or     a
DFDB  C9          ret

******************************************* Befehle mit Zeilennummer
DFDC  C7          db     C7'          RESTORE'
DFDD  81          db     81'          AUTO'
DFDE  C6          db     C6'          RENUM'
DFDF  92          db     92'          DELETE'
DFE0  96          db     96'          EDIT'
DFE1  C8          db     C8'          RESUME'
DFE2  E3          db     E3'          ERL'
DFE3  97          db     97'          ELSE'
DFE4  CA          db     CA'          RUN'
DFE5  A7          db     A7'          LIST'
DFE6  A0          db     A0'          GOTO'
DFE7  EB          db     EB'          THEN'
DFE8  9F          db     9F'          GOSUB'
DFE9  00          db     00           Ende der Tabelle

*******************************************************
DFEA  FE26        cp     26           '&'
DFEC  D0          ret    nc
DFED  FE21        cp     21           '!'
DFF0  D0          ret    nc
DFF1  FE22        cp     22           '"'
DFF3  C8          ret    z
DFF4  FE23        cp     23           '#'
DFF6  C8          ret    z
DFF7  EE27        xor    27           '''
DFF9  FE04        cp     04
DFFB  CEFF        adc    a,FF
DFFD  37          scf
DFFE  c9          ret
```

```
E000  39       add   hl,sp
E001  AE       xor   (hl)
E002  B7       or    a
E003  2815     jr    z,E01A
E005  7E       ld    a,(hl)
E006  23       inc   hl
E007  FA25DF   jp    m,DF25      in Puffer schreiben
E00A  FE2E     cp    2E          '.'
E00C  CA25DF   jp    z,DF25      in Puffer schreiben
E00F  2B       dec   hl
E010  D5       push  de
E011  CD04EE   call  EE04
E014  3034     jr    nc,E04A
E016  3E1E     ld    a,1E        Token für Zeilennummer
E018  184F     jr    E069

E01A  D5       push  de
E01B  C5       push  bc
E01C  CDBEEC   call  ECBE
E01F  C1       pop   bc
E020  3028     jr    nc,E04A
E022  CD27FF   call  FF27        Test auf String
E025  3E1F     ld    a,1F        Token für Fließkomma
E027  3040     jr    nc,E069
E029  EB       ex    de,hl
E02A  2AC2B0   ld    hl,(B0C2)
E02D  EB       ex    de,hl
E02E  7A       ld    a,d
E02F  B7       or    a
E030  3E1A     ld    a,1A        Token für Zwei-Byte-Zahl
E032  2035     jr    nz,E069
E034  E3       ex    (sp),hl
E035  EB       ex    de,hl
E036  7D       ld    a,l
E037  FE0A     cp    0A          10
E039  3004     jr    nc,E03F
E03B  C60E     add   a,0E        Offset addieren
E03D  1806     jr    E045

E03F  3E19     ld    a,19        Token für Ein-Byte-Wert
E041  CD25DF   call  DF25        in Puffer schreiben
E044  7D       ld    a,l
E045  CD25DF   call  DF25        in Puffer schreiben
E048  E1       pop   hl
E049  C9       ret

E04A  7E       ld    a,(hl)
E04B  23       inc   hl
E04C  E3       ex    (sp),hl
E04D  EB       ex    de,hl
E04E  CD25DF   call  DF25        in Puffer schreiben
E051  EB       ex    de,hl
E052  E3       ex    (sp),hl
E053  CDB8FF   call  FFB8        Vergleich hl <> de
E056  20F2     jr    nz,E04A
```

```
E058  D1        pop   de
E059  C9        ret

E05A  D5        push  de
E05B  C5        push  bc
E05C  CDBEEC    call  ECBE
E05F  C1        pop   bc
E060  30E8      jr    nc,E04A
E062  FE02      cp    02
E064  3E1B      ld    a,1B          Token für Binärzahl
E066  2801      jr    z,E069
E068  3C        inc   a
E069  D1        pop   de
E06A  CD25DF    call  DF25          in Puffer schreiben
E06D  E5        push  hl
E06E  21C2B0    ld    hl,B0C2
E071  CD23FF    call  FF23          Variablentyp holen
E074  F5        push  af
E075  7E        ld    a,(hl)
E076  23        inc   hl
E077  CD25DF    call  DF25          in Puffer schreiben
E07A  F1        pop   af
E07B  3D        dec   a
E07C  20F6      jr    nz,E074
E07E  E1        pop   hl
E07F  C9        ret

E080  FE22      cp    22            '"'
E082  283B      jr    z,E0BF
E084  FE7C      cp    7C            '|', Befehlserweiterung
E086  2845      jr    z,E0CD
E088  C5        push  bc
E089  D5        push  de
E08A  EE3F      xor   3F            '?'
E08C  06BF      ld    b,BF          'PRINT'
E08E  2816      jr    z,E0A6
E090  2B        dec   hl
E091  114BE6    ld    de,E64B       Adresse der BASIC-Operatoren
E094  CD27E3    call  E327
E097  1A        ld    a,(de)
E098  3808      jr    c,E0A2
E09A  7E        ld    a,(hl)
E09B  FE20      cp    20            ' '
E09D  3002      jr    nc,E0A1
E09F  3E20      ld    a,20          ' '
E0A1  23        inc   hl
E0A2  47        ld    b,a
E0A3  CDB3E0    call  E0B3
E0A6  3239AE    ld    (AE39),a
E0A9  78        ld    a,b
E0AA  D1        pop   de
E0AB  C1        pop   bc
E0AC  FEC0      cp    C0            '''
E0AE  2836      jr    z,E0E6
```

```
E0B0  C325DF      jp     DF25         in Puffer schreiben
E0B3  3D          dec    a
E0B4  C8          ret    z
E0B5  EE22        xor    22           ’“’
E0B7  C8          ret    z
E0B8  3A39AE      ld     a,(AE39)
E0BB  3C          inc    a
E0BC  C8          ret    z
E0BD  3D          dec    a
E0BE  C9          ret

E0BF  CD25DF      call   DF25         in Puffer schreiben
E0C2  7E          ld     a,(hl)
E0C3  B7          or     a
E0C4  C8          ret    z
E0C5  23          inc    hl
E0C6  FE22        cp     22           ’“’
E0C8  20F5        jr     nz,E0BF
E0CA  C325DF      jp     DF25         in Puffer schreiben

***************************************** Befehlserweiterung verarbeiten
E0CD  CD25DF      call   DF25         in Puffer schreiben
E0D0  AF          xor    a            Null
E0D1  3239AE      ld     (AE39),a
E0D4  CD25DF      call   DF25         in Puffer schreiben
E0D7  7E          ld     a,(hl)       nächstes Zeichen
E0D8  23          inc    hl           Zeiger erhöhen
E0D9  CD7BFF      call   FF7B         Test auf Buchstabe oder Ziffer
E0DC  38F6        jr     c,E0D4       ja, dann in Puffer
E0DE  2B          dec    hl
E0DF  1B          dec    de           Zeiger eins zurück
E0E0  1A          ld     a,(de)
E0E1  F680        or     80           beim letzten Zeichen Bit 7 setzen
E0E3  12          ld     (de),a
E0E4  13          inc    de
E0E5  C9          ret

E0E6  3E01        ld     a,01
E0E8  CD25DF      call   DF25         in Puffer schreiben
E0EB  3EC0        ld     a,C0         ’’’
E0ED  CD25DF      call   DF25         in Puffer schreiben
E0F0  7E          ld     a,(hl)       Zeichen
E0F1  23          inc    hl
E0F2  B7          or     a            bis Zeilenende
E0F3  20F8        jr     nz,E0ED      in Puffer schreiben
E0F5  2B          dec    hl
E0F6  C9          ret

************************************************** BASIC-Befehl LIST
E0F7  CDB0CE      call   CEB0         Zeilennummernbereich holen
E0FA  C5          push   bc
E0FB  D5          push   de
```

```
E0FC  CDC6C1    call  C1C6      Kanalnummer holen
E0FF  CD4ADD    call  DD4A      Ende des Statements, sonst 'Syntax error'
E102  CDCBDD    call  DDCB      aktuelle Zeilenadresse auf Null
E105  D1        pop   de
E106  C1        pop   bc
E107  CD0DE1    call  E10D      Zeilen listen
E10A  C364C0    jp    C064      zum READY-Modus

******************************************* BASIC-Zeilen bc -de listen
E10D  D5        push  de
E10E  50        ld    d,b       Zeilennummer nach de
E10F  59        ld    e,c
E110  CDA3E7    call  E7A3      BASIC-Zeile de suchen
E113  D1        pop   de
E114  4E        ld    c,(hl)
E115  23        inc   hl        Programmende ?
E116  46        ld    b,(hl)
E117  2B        dec   hl
E118  78        ld    a,b
E119  B1        or    c
E11A  C8        ret   z         fertig
E11B  CD3CC4    call  C43C      Unterbrechung durch 'ESC' ?
E11E  E5        push  hl
E11F  09        add   hl,bc     Zeilenlänge addieren
E120  E3        ex    (sp),hl
E121  D5        push  de
E122  E5        push  hl
E123  23        inc   hl
E124  23        inc   hl
E125  5E        ld    e,(hl)
E126  23        inc   hl        nächste Zeilennummer nach de
E127  56        ld    d,(hl)
E128  E1        pop   hl
E129  E3        ex    (sp),hl
E12A  CDB8FF    call  FFB8      Vergleich hl <> de
E12D  E3        ex    (sp),hl
E12E  3812      jr    c,E142    größer letzte Zeilennummer ?
E130  CD63E1    call  E163      BASIC-Zeile in Puffer listen
E133  CD45E1    call  E145      Zeichen aus Puffer ausgeben
E136  23        inc   hl
E137  7E        ld    a,(hl)
E138  B7        or    a         letztes Zeichen ?
E139  20F8      jr    nz,E133   nein
E13B  CD4EC3    call  C34E      LF ausgeben
E13E  D1        pop   de
E13F  E1        pop   hl
E140  18D2      jr    E114

E142  E1        pop   hl
E143  E1        pop   hl
E144  C9        ret
```

```
*************************************** Zeichen aus Puffer ausgeben
E145  CDBAC1     call   C1BA        Ausgabekanal kleiner 8 ?
E148  380B       jr     c,E155      ja, Bildschirmausgabe
E14A  7E         ld     a,(hl)
E14B  CD6EC3     call   C36E        Zeichen ausgeben
E14E  FE0A       cp     0A          LF ?
E150  C0         ret    nz
E151  3E0D       ld     a,0D        CR hinterher schicken
E153  180B       jr     E160

******************************************** Bildschirmausgabe
E155  7E         ld     a,(hl)      Zeichen holen
E156  FE20       cp     20          Kontrollzeichen ?
E158  3006       jr     nc,E160     nein, so ausgeben
E15A  3E01       ld     a,01        Kontrollzeichen als druckbare Zeichen
E15C  CD6EC3     call   C36E        Zeichen ausgeben
E15F  7E         ld     a,(hl)      Zeichen holen
E160  C36EC3     jp     C36E        und ausgeben

*************************************** BASIC-Zeile in Puffer listen
E163  D5         push   de
E164  01A4AC     ld     bc,ACA4     Zeiger auf Eingabepuffer
E167  C5         push   bc
E168  23         inc    hl
E169  23         inc    hl
E16A  5E         ld     e,(hl)
E16B  23         inc    hl          Zeilennummer nach de
E16C  56         ld     d,(hl)
E16D  23         inc    hl
E16E  E5         push   hl
E16F  EB         ex     de,hl
E170  CD0DFF     call   FF0D        Integerzahl hl übernehmen
E173  CD82EE     call   EE82        nach ASCII wandeln
E176  110000     ld     de,0000
E179  7E         ld     a,(hl)
E17A  23         inc    hl
E17B  B7         or     a
E17C  2805       jr     z,E183
E17E  CDFEE1     call   E1FE        in Puffer schreiben
E181  18F6       jr     E179

E183  3E20       ld     a,20        ' '
E185  CDFEE1     call   E1FE        in Puffer schreiben
E188  E1         pop    hl
E189  7E         ld     a,(hl)      Zeichen aus Programm holen
E18A  B7         or     a
E18B  2805       jr     z,E192      Zeilenende ?
E18D  CD96E1     call   E196        Token expandieren
E190  18F7       jr     E189

E192  02         ld     (bc),a
E193  E1         pop    hl
```

```
E194  D1            pop   de
E195  C9            ret

E196  CD13AC        call  AC13        ret
E199  FA20E2        jp    m,E220      Befehlstoken ?
E19C  FE02          cp    02
E19E  381D          jr    c,E1BD
E1A0  FE05          cp    05
E1A2  3843          jr    c,E1E7
E1A4  FE0B          cp    0B
E1A6  3822          jr    c,E1CA
E1A8  FE0E          cp    0E
E1AA  383B          jr    c,E1E7
E1AC  FE20          cp    20          ' '
E1AE  382E          jr    c,E1DE      Konstante ausgeben
E1B0  FE7C          cp    7C          '|', Befehlserweiterung
E1B2  2851          jr    z,E205
E1B4  CDEADF        call  DFEA
E1B7  DC1AE2        call  c,E21A      Blank ausgeben
E1BA  7E            ld    a,(hl)
E1BB  180D          jr    E1CA

E1BD  23            inc   hl
E1BE  7E            ld    a,(hl)
E1BF  FEC0          cp    C0          '''
E1C1  285D          jr    z,E220
E1C3  FE97          cp    97          'ELSE'
E1C5  2859          jr    z,E220
E1C7  2B            dec   hl
E1C8  3E3A          ld    a,3A        ':'
E1CA  1E00          ld    e,00
E1CC  FE22          cp    22          '"'
E1CE  200B          jr    nz,E1DB
E1D0  CDFEE1        call  E1FE        in Puffer schreiben
E1D3  23            inc   hl
E1D4  7E            ld    a,(hl)
E1D5  B7            or    a
E1D6  C8            ret   z
E1D7  FE22          cp    22          '"'
E1D9  20F5          jr    nz,E1D0
E1DB  23            inc   hl
E1DC  1820          jr    E1FE        in Puffer schreiben

E1DE  CD1AE2        call  E21A        Blank ausgeben
E1E1  CD53E2        call  E253        Konstante ausgeben
E1E4  1E01          ld    e,01
E1E6  C9            ret

E1E7  CD1AE2        call  E21A        Blank ausgeben
E1EA  7E            ld    a,(hl)
E1EB  F5            push  af
E1EC  23            inc   hl
```

```
E1ED 23        inc   hl
E1EE 23        inc   hl
E1EF CD0FE2    call  E20F
E1F2 F1        pop   af
E1F3 1E01      ld    e,01
E1F5 FE0B      cp    0B
E1F7 D0        ret   nc
E1F8 1E00      ld    e,00
E1FA EE27      xor   27
E1FC E6FD      and   FD
E1FE 02        ld    (bc),a       Zeichen in Puffer schreiben
E1FF 03        inc   bc           Pufferzeiger erhöhen
E200 15        dec   d
E201 C0        ret   nz
E202 0B        dec   bc
E203 14        inc   d
E204 C9        ret

*************************************** Befehlserweiterung listen
E205 1E01      ld    e,01
E207 CDFEE1    call  E1FE         in Puffer schreiben
E20A 23        inc   hl
E20B 7E        ld    a,(hl)       nächstes Zeichen
E20C 23        inc   hl           Zeiger erhöhen
E20D B7        or    a
E20E C0        ret   nz           Zeilenende ?
E20F 7E        ld    a,(hl)       Zeichen holen
E210 E67F      and   7F           Bit 7 löschen
E212 CDFEE1    call  E1FE         in Puffer schreiben
E215 BE        cp    (hl)         letztes Zeichen ?
E216 23        inc   hl
E217 30F6      jr    nc,E20F      nein, nächstes Zeichen
E219 C9        ret

E21A 1D        dec   e
E21B C0        ret   nz
E21C 3E20      ld    a,20         ' '
E21E 18DE      jr    E1FE         in Puffer schreiben

E220 23        inc   hl
E221 FEFF      cp    FF           Funktion ?
E223 2002      jr    nz,E227      nein
E225 7E        ld    a,(hl)
E226 23        inc   hl
E227 F5        push  af
E228 E5        push  hl
E229 CDEDE2    call  E2ED         Token listen ?
E22C B7        or    a
E22D 2808      jr    z,E237
E22F F5        push  af
E230 CD1AE2    call  E21A
E233 F1        pop   af
```

```
E234  CDFEE1     call  E1FE       in Puffer schreiben
E237  7E         ld    a,(hl)
E238  E67F       and   7F
E23A  FE09       cp    09
E23C  C4FEE1     call  nz,E1FE    in Puffer schreiben
E23F  BE         cp    (hl)
E240  23         inc   hl
E241  28F4       jr    z,E237
E243  CD7BFF     call  FF7B       Test auf Buchstabe oder Ziffer
E246  1E00       ld    e,00
E248  3002       jr    nc,E24C
E24A  1E01       ld    e,01
E24C  E1         pop   hl
E24D  F1         pop   af
E24E  D6E4       sub   E4
E250  C0         ret   nz
E251  5F         ld    e,a
E252  C9         ret

E253  D5         push  de
E254  7E         ld    a,(hl)
E255  23         inc   hl
E256  FE1B       cp    1B         Binärzahl ?
E258  2849       jr    z,E2A3
E25A  FE1C       cp    1C         Hexzahl ?
E25C  2850       jr    z,E2AE
E25E  FE1E       cp    1E         Zeilenadresse ?
E260  2826       jr    z,E288
E262  FE1D       cp    1D         Zeilennummer ?
E264  2822       jr    z,E288
E266  FE1F       cp    1F         Fließkommazahl ?
E268  285E       jr    z,E2C8
E26A  FE19       cp    19         Ein-Byte-Zahl ?
E26C  2809       jr    z,E277
E26E  FE1A       cp    1A         Zwei-Byte-Zahl ?
E270  280B       jr    z,E27D
E272  D60E       sub   0E         Ziffer ?
E274  5F         ld    e,a
E275  1802       jr    E279

******************************************** Ein-Byte-Zahl ausgeben
E277  5E         ld    e,(hl)     Lo-byte
E278  23         inc   hl
E279  1600       ld    d,00       Hi-Byte Null
E27B  1804       jr    E281

******************************************** Zwei-Byte-Zahl ausgeben
E27D  5E         ld    e,(hl)     Lo-Byte
E27E  23         inc   hl
E27F  56         ld    d,(hl)     Hi-Byte
E280  23         inc   hl
E281  E3         ex    (sp),hl
```

```
E282  EB          ex     de,hl
E283  CD0DFF      call   FF0D          Integerzahl hl übernehmen
E286  1847        jr     E2CF

*******************************************  Zeilennummer ausgeben
E288  5E          ld     e,(hl)
E289  23          inc    hl
E28A  56          ld     d,(hl)
E28B  23          inc    hl
E28C  FE1E        cp     1E            Zeilennummer ?
E28E  2809        jr     z,E299        ja
E290  E5          push   hl
E291  EB          ex     de,hl
E292  23          inc    hl
E293  23          inc    hl
E294  23          inc    hl
E295  5E          ld     e,(hl)
E296  23          inc    hl
E297  56          ld     d,(hl)
```

```
E298  E1            pop    hl
E299  E3            ex     (sp),hl
E29A  EB            ex     de,hl
E29B  CD0DFF        call   FF0D          Integerzahl hl übernehmen
E29E  CD82EE        call   EE82          nach ASCII wandeln
E2A1  182F          jr     E2D2

********************************************** Binärzahl ausgeben
E2A3  C5            push   bc
E2A4  010200        ld     bc,0002
E2A7  CD14F1        call   F114          in Binärzahl wandeln
E2AA  3E58          ld     a,58          'X'
E2AC  1809          jr     E2B7

********************************************** Hexzahl ausgeben
E2AE  C5            push   bc
E2AF  010200        ld     bc,0002
E2B2  CD19F1        call   F119          in Hexzahl wandeln
E2B5  3E48          ld     a,48          'H'
E2B7  C1            pop    bc
E2B8  E3            ex     (sp),hl
E2B9  EB            ex     de,hl
E2BA  F5            push   af
E2BB  3E26          ld     a,26          '&'
E2BD  CDFEE1        call   E1FE          in Puffer schreiben
E2C0  F1            pop    af
E2C1  FE48          cp     48            'H' nicht
E2C3  C4FEE1        call   nz,E1FE       in Puffer schreiben
E2C6  180A          jr     E2D2

******************************************** Fließkommazahl ausgeben
E2C8  3E05          ld     a,05          Variablentyp 'Real'
E2CA  CD4BFF        call   FF4B          Zahl holen
E2CD  E3            ex     (sp),hl
E2CE  EB            ex     de,hl
E2CF  CD8FEE        call   EE8F          nach ASCII wandeln
E2D2  7E            ld     a,(hl)        Zeichen holen
E2D3  23            inc    hl
E2D4  CDFEE1        call   E1FE          in Puffer schreiben
E2D7  7E            ld     a,(hl)
E2D8  B7            or     a             Ende der Zahl ?
E2D9  20F7          jr     nz,E2D2       nein
E2DB  E1            pop    hl
E2DC  C9            ret

*************************************************************
E2DD  E5            push   hl
E2DE  D641          sub    41            'A'
E2E0  87            add    a,a
E2E1  C654          add    a,54
E2E3  6F            ld     l,a           plus E354, Adressen der Befehlsworte
E2E4  CEE3          adc    a,E3
E2E6  95            sub    l
E2E7  67            ld     h,a
E2E8  5E            ld     e,(hl)
```

```
E2E9  23          inc   hl
E2EA  56          ld    d,(hl)
E2EB  E1          pop   hl
E2EC  C9          ret

E2ED  C5          push  bc
E2EE  4F          ld    c,a
E2EF  061A        ld    b,1A          26 Buchstaben
E2F1  2188E3      ld    hl,E388       Tabelle der Befehlsworte
E2F4  CD13E3      call  E313
E2F7  380D        jr    c,E306
E2F9  23          inc   hl
E2FA  10F8        djnz  E2F4          nächster Buchstabe
E2FC  214BE6      ld    hl,E64B       Tabelle der BASIC-Operatoren
E2FF  CD13E3      call  E313
E302  3007        jr    nc,E30B
E304  06C0        ld    b,C0
E306  78          ld    a,b
E307  C640        add   a,40
E309  C1          pop   bc
E30A  C9          ret

E30B  CD19AC      call  AC19          ret
E30E  1E02        ld    e,02          'Syntax error'
E310  C394CA      jp    CA94          Fehlermeldung ausgeben

E313  7E          ld    a,(hl)
E314  B7          or    a
E315  C8          ret   z
E316  E5          push  hl
E317  7E          ld    a,(hl)
E318  23          inc   hl
E319  17          rla
E31A  30FB        jr    nc,E317
E31C  7E          ld    a,(hl)
E31D  23          inc   hl
E31E  B9          cp    c
E31F  2803        jr    z,E324
E321  F1          pop   af
E322  18EF        jr    E313

E324  E1          pop   hl
E325  37          scf
E326  C9          ret

E327  1A          ld    a,(de)
E328  B7          or    a
E329  C8          ret   z
E32A  E5          push  hl
E32B  1A          ld    a,(de)
E32C  13          inc   de
E32D  FE09        cp    09            TAB ?
E32F  2804        jr    z,E335
E331  FE20        cp    20            ' '
E333  2005        jr    nz,E33A
```

```
E335  CD61DD        call   DD61           Blank, TAB und LF überlesen
E338  18F1          jr     E32B

E33A  4F            ld     c,a
E33B  7E            ld     a,(hl)
E33C  23            inc    hl
E33D  CD8AFF        call   FF8A           Klein- in Großbuchstaben wandeln
E340  A9            xor    c
E341  28E8          jr     z,E32B
E343  E67F          and    7F
E345  280A          jr     z,E351
E347  1B            dec    de
E348  1A            ld     a,(de)
E349  13            inc    de
E34A  17            rla
E34B  30FB          jr     nc,E348
E34D  13            inc    de
E34E  E1            pop    hl
E34F  18D6          jr     E327

E351  F1            pop    af
E352  37            scf
E353  C9            ret

******************************************** Adressen der Befehlsworte
E554  35E6          dw     E635           A
E356  2AE6          dw     E62A           B
E358  EFE5          dw     E5EF           C
E35A  B9E5          dw     E5B9           D
E35C  8AE5          dw     E58A           E
E35E  7EE5          dw     E57E           F
E360  72E5          dw     E572           G
E362  68E5          dw     E568           H
E364  47E5          dw     E547           I
E366  43E5          dw     E543           J
E368  3FE5          dw     E53F           K
E36A  13E5          dw     E513           L
E36C  EDE4          dw     E4ED           M
E36E  E2E4          dw     E4E2           N
E370  AAE4          dw     E4AA           O
E372  86E4          dw     E486           P
E374  85E4          dw     E485           Q
E376  3BE4          dw     E43B           R
E378  FBE3          dw     E3FB           S
E37A  CFE3          dw     E3CF           T
E37C  C0E3          dw     E3C0           U
E37E  B8E3          dw     E3B8           V
E380  9AE3          dw     E39A           W
E382  92E3          dw     E392           X
E384  8DE3          dw     E38D           Y
E386  88E3          dw     E388           Z

******************************************** Tabelle der BASIC-Befehle
```

```
********************************************************** Buchstabe Z
E388  4F  4E  C5                              DA ZONE
E38C  00

********************************************************** Buchstabe Y
E38D  50  4F  D3                              48 YPOS
E391  00

********************************************************** Buchstabe X
E392  50  4F  D3                              47 XPOS
E396  4F  D2                                  FD XOR
E399  00

********************************************************** Buchstabe W
E39A  52  49  54  C5                          D9 WRITE
E39F  49  4E  44  4F  D7                      D8 WINDOW
E3A5  49  44  54  C8                          D7 WIDTH
E3AA  48  49  4C  C5                          D6 WHILE
E3AF  45  4E  C4                              D5 WEND
E3B3  41  49  D4                              D4 WAIT
E3B7  00

********************************************************** Buchstabe V
E3B8  50  4F  D3                              7F VPOS
E3BC  41  CC                                  1D VAL
E3BF  00

********************************************************** Buchstabe U
E3C0  53  49  4E  C7                          ED USING
E3C5  50  50  45  52  A4                      1C UPPER$
E3CB  4E  D4                                  1B UNT
E3CE  00

********************************************************** Buchstabe T
E3CF  52  4F  CE                              D3 TRON
E3D3  52  4F  46  C6                          D2 TROFF
E3D8  CF                                      EC TO
E3DA  49  4D  C5                              46 TIME
E3DE  48  45  CE                              EB THEN
E3E2  45  53  54  D2                          7D TESTR
E3E7  45  53  D4                              7C TEST
E3EB  41  CE                                  1A TAN
E3EE  41  47  4F  46  C6                      D1 TAGOFF
E3F4  41  C7                                  D0 TAG
E3F7  41  C2                                  EA TAB
E3FA  00

********************************************************** Buchstabe S
E3FB  59  4D  42  4F  CC                      CF SYMBOL
E401  57  41  D0                              E7 SWAP
E405  54  52  49  4E  47  A4                  7B STRING$
E40C  54  52  A4                              19 STR$
E410  54  4F  D0                              CE STOP
E414  54  45  D0                              E6 STEP
E418  51  D2                                  18 SQR
```

```
E41B  D1                              17 SQ
E41D  50 45 45 C4                     CD SPEED
E422  50 C3                           E5 SPC
E425  50 41 43 45 A4                  16 SPACE$
E42B  4F 55 4E C4                     CC SOUND
E430  49 CE                           15 SIN
E433  47 CE                           14 SGN
E436  41 56 C5                        CB SAVE
E43A  00

******************************************************** Buchstabe R
E43B  55 CE                           CA RUN
E43E  4F 55 4E C4                     7A ROUND
E443  4E C4                           45 RND
E446  49 47 48 54 A4                  79 RIGHT$
E44C  45 54 55 52 CE                  C9 RETURN
E452  45 53 55 4D C5                  C8 RESUME
E458  45 53 54 4F 52 C5               C7 RESTORE
E45F  45 4E 55 CD                     C6 RENUM
E464  45 4D 41 49 CE                  13 REMAIN
E46A  45 CD                           C5 REM
E46D  45 4C 45 41 53 C5               C4 RELEASE
E474  45 41 C4                        C3 READ
E478  41 4E 44 4F 4D 49 5A C5         C2 RANDOMIZE
E481  41 C4                           C1 RAD
E484  00

******************************************************** Buchstabe Q
E485  00

******************************************************** Buchstabe P
E486  52 49 4E D4                     BF PRINT
E48B  4F D3                           78 POS
E48E  4F 4B C5                        BE POKE
E492  4C 4F 54 D2                     BD PLOTR
E497  4C 4F D4                        BC PLOT
E49B  C9                              44 PI
E49D  45 CE                           BB PEN
E4A0  45 45 CB                        12 PEEK
E4A4  41 50 45 D2                     BA PAPER
E4A9  00

******************************************************** Buchstabe O
E4AA  55 D4                           B9 OUT
E4AD  52 49 47 49 CE                  B8 ORIGIN
E4B3  D2                              FC OR
E4B5  50 45 4E 4F 55 D4               B7 OPENOUT
E4BC  50 45 4E 49 CE                  B6 OPENIN
E4C2  4E 20 53 D1                     B5 ON SQ
E4C7  4E 20 45 52 52 4F 52 20
E4CF  47 4F 09 54 4F 20 B0            B4 ON ERROR GO TO 0
E4D7  4E 20 42 52 45 41 CB            B3 ON BREAK
E4DF  CE                              B2 ON
E4E1  00
```

```
********************************************************* Buchstabe N
E4E2  4F D4                        FE NOT
E4E5  45 D7                        B1 NEW
E4E8  45 58 D4                     B0 NEXT
E4EC  00

********************************************************* Buchstabe M
E4ED  4F 56 45 D2                  AF MOVER
E4F2  4F 56 C5                     AE MOVE
E4F6  4F 44 C5                     AD MODE
E4FA  4F C4                        FB MOD
E4FD  49 CE                        77 MIN
E500  49 44 A4                     AC MID$
E504  45 52 47 C5                  AB MERGE
E509  45 4D 4F 52 D9               AA MEMORY
E50F  41 D8                        76 MAX
E512  00

********************************************************* Buchstabe L
E513  4F 57 45 52 A4               11 LOWER$
E519  4F 47 31 B0                  10 LOG10
E51E  4F C7                        0F LOG
E521  4F 43 41 54 C5               A9 LOCATE
E527  4F 41 C4                     A8 LOAD
E52B  49 53 D4                     A7 LIST
E52F  49 4E C5                     A6 LINE
E533  45 D4                        A5 LET
E536  45 CE                        0E LEN
E539  45 46 54 A4                  75 LEFT$
E53E  00

********************************************************* Buchstabe K
E53F  45 D9                        A4 KEY
E542  00

********************************************************* Buchstabe J
E543  4F D9                        0D JOY
E546  00

********************************************************* Buchstabe I
E547  4E D4                        0C INT
E54A  4E 53 54 D2                  74 INSTR
E54F  4E 50 55 D4                  A3 INPUT
E554  4E D0                        0B INP
E557  4E 4B 45 59 A4               43 INKEY$
E55D  4E 4B 45 D9                  0A INKEY
E562  4E CB                        A2 INK
E565  C6                           A1 IF
E567  00

********************************************************* Buchstabe H
E568  49 4D 45 CD                  42 HIMEM
E56D  45 58 A4                     73 HEX$
E571  00
```

```
************************************************ Buchstabe G
E572  4F 09 54 CF                       A0 GO TO
E577  4F 09 53 55 C2                    9F GO SUB
E57D  00

************************************************ Buchstabe F
E57E  52 C5                             09 FRE
E581  4F D2                             9E FOR
E584  CE                                E4 FN
E586  49 D8                             08 FIX
E589  00

************************************************ Buchstabe E
E58A  58 D0                             07 EXP
E58D  56 45 52 D9                       9D EVERY
E592  52 52 4F D2                       9C ERROR
E597  52 D2                             41 ERR
E59A  52 CC                             E3 ERL
E59D  52 41 53 C5                       9B ERASE
E5A2  4F C6                             40 EOF
E5A5  4E D6                             9A ENV
E5A8  4E D4                             99 ENT
E5AB  4E C4                             98 END
E5AE  4C 53 C5                          97 ELSE
E5B2  C9                                DC EI
E5B4  44 49 D4                          96 EDIT
E5B8  00

************************************************ Buchstabe D
E5B9  52 41 57 D2                       95 DRAWR
E5BE  52 41 D7                          94 DRAW
E5C2  49 CD                             93 DIM
E5C5  C9                                DB DI
E5C7  45 4C 45 54 C5                    92 DELETE
E5CD  45 C7                             91 DEG
E5D0  45 46 53 54 D2                    90 DEFSTR
E5D6  45 46 52 45 41 CC                 8F DEFREAL
E5DD  45 46 49 4E D4                    8E DEFINT
E5E3  45 C6                             8D DEF
E5E6  45 43 A4                          72 DEC$
E5EA  41 54 C1                          8C DATA
E5EE  00

************************************************ Buchstabe C
E5EF  52 45 41 CC                       06 CREAL
E5F4  4F D3 OS 05                       05 COS
E5F7  4F 4E D4                          8B CONT
E5FB  4C D3                             8A CLS
E5FE  4C 4F 53 45 4F 55 D4              89 CLOSEOUT
E606  4C 4F 53 45 49 CE                 88 CLOSEIN
E60D  4C C7                             87 CLG
E610  4C 45 41 D2                       86 CLEAR
E615  49 4E D4                          04 CINT
E619  48 52 A4                          03 CHR$
E61D  48 41 49 CE                       85 CHAIN
```

```
E622  41  D4                                          84 CAT
E625  41  4C  CC                                      83 CALL
E629  00

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  Buchstabe B
E62A  4F  52  44  45  D2                              82 BORDER
E630  49  4E  A4                                      71 BIN$
E634  00

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  Buchstabe A
E635  55  54  CF                                      81 AUTO
E639  54  CE                                          02 ATN
E63C  53  C3                                          01 ASC
E63F  4E  C4                                          FA AND
E642  46  54  45                                      80 AFTER
E647  42  D3                                          00 ABS
E64A  00


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  BASIC-Operatoren und zugehörige Token
E64B  DE              db    DE              '↑'
E64C  F8              db    F8
E64D  DC              db    DC              'Backslash'
E64E  F9              db    F9
E64F  3E09BD          db    3E,09,BD        '>='
E652  F0              db    F0
E653  3D20BE          db    3D,20,BE        '=>'
E656  F0              db    F0
E657  BE              db    BE              '>'
E658  EE              db    EE
E659  BD              db    BD              '='
E65A  EF              db    EF
E65B  3C09BE          db    3C,09,BE        '<>'
E65E  F2              db    F2
E65F  3C09BD          db    3C,09,BD        '<='
E662  F3              db    F3
E663  3D20BC          db    3D,20,BC        '=<'
E666  F3              db    F3
E667  BC              db    BC              '<'
E668  F1              db    F1
E669  AF              db    AF              '/'
E66A  F7              db    F7
E66B  BA              db    BA              ':'
E66C  01              db    01
E66D  AA              db    AA              '*'
E66E  F6              db    F6
E66F  AD              db    AD              '-'
E670  F5              db    F5
E671  AB              db    AB              '+'
E672  F4              db    F4
E673  A7              db    A7              '''
E674  C0              db    C0
E675  00              db    00
```

```
************************************************ Programmzeiger löschen
E676  AF            xor    a
E677  323AAE        ld     (AE3A),a
E67A  2A81AE        ld     hl,(AE81)        Programmstart
E67D  77            ld     (hl),a
E67E  23            inc    hl
E67F  77            ld     (hl),a           dreimal Null ans Programmende
E680  23            inc    hl
E681  77            ld     (hl),a
E682  23            inc    hl
E683  2283AE        ld     (AE83),hl        Programmende
E686  C9            ret

********************************************************************
E687  3A3AAE        ld     a,(AE3A)
E68A  B7            or     a
E68B  C8            ret    z
E68C  C5            push   bc
E68D  D5            push   de
E68E  E5            push   hl
E68F  019DE6        ld     bc,E69D          Zeilennummern einsetzen
E692  CDFFE8        call   E8FF
E695  AF            xor    a
E696  323AAE        ld     (AE3A),a
E699  E1            pop    hl
E69A  D1            pop    de
E69B  C1            pop    bc
E69C  C9            ret

******************************** Zeilenadressen durch Zeilennummer ersetzen
E69D  CD43E9        call   E943             nächstes Element der Zeile holen
E6A0  FE02          cp     02               Ende des Statements ?
E6A2  D8            ret    c                ja
E6A3  FE1D          cp     1D               'Zeilenadresse' ?
E6A5  20F6          jr     nz,E69D          nein
E6A7  56            ld     d,(hl)
E6A8  2B            dec    hl
E6A9  5E            ld     e,(hl)
E6AA  2B            dec    hl
E6AB  E5            push   hl
E6AC  EB            ex     de,hl
E6AD  23            inc    hl
E6AE  23            inc    hl
E6AF  23            inc    hl
E6B0  5E            ld     e,(hl)
E6B1  23            inc    hl               Zeilennummer nach de
E6B2  56            ld     d,(hl)
E6B3  E1            pop    hl
E6B4  361E          ld     (hl),1E          'Zeilennummer'
E6B6  23            inc    hl
E6B7  73            ld     (hl),e
E6B8  23            inc    hl               einsetzen
E6B9  72            ld     (hl),d
E6BA  18E1          jr     E69D
```

```
************************************** Eingabezeile in Interpreterkode wandeln
E6BC CD61DD     call  DD61       Blank, TAB und LF überlesen
E6BF B7         or    a
E6C0 37         scf
E6C1 C8         ret   z
E6C2 CD04EE     call  EE04
E6C5 D0         ret   nc
E6C6 7E         ld    a,(hl)
E6C7 FE20       cp    20         ' '
E6C9 2001       jr    nz,E6CC
E6CB 23         inc   hl
E6CC CDD2E6     call  E6D2       Statement in Interpreterkode wandeln
E6CF 37         scf
E6D0 9F         sbc   a,a
E6D1 C9         ret

*************************************** Statement in Interpreterkode wandeln
E6D2 CD87E6     call  E687
E6D5 CDBBDE     call  DEBB       Token in Puffer ab (&AE7F) (&40)
E6D8 E5         push  hl
E6D9 CD61DD     call  DD61       Blank, TAB und LF überlesen
E6DC B7         or    a
E6DD 2828       jr    z,E707
E6DF C5         push  bc
E6E0 D5         push  de
E6E1 210400     ld    hl,0004
E6E4 09         add   hl,bc
E6E5 E5         push  hl
E6E6 E5         push  hl
E6E7 CDA3E7     call  E7A3       BASIC-Zeile de suchen
E6EA E5         push  hl
E6EB DC0BE7     call  c,E70B
E6EE D1         pop   de
E6EF C1         pop   bc         Anzahl Bytes
E6F0 CDF8F5     call  F5F8       Platz im Variablenbereich reservieren
E6F3 CD2CF5     call  F52C       Prg- und Variablenpointer um bc erhöhen
E6F6 EB         ex    de,hl
E6F7 D1         pop   de
E6F8 73         ld    (hl),e
E6F9 23         inc   hl
E6FA 72         ld    (hl),d
E6FB 23         inc   hl
E6FC D1         pop   de
E6FD 73         ld    (hl),e
E6FE 23         inc   hl
E6FF 72         ld    (hl),d
E700 23         inc   hl
E701 C1         pop   bc
E702 EB         ex    de,hl
E703 E1         pop   hl
E704 C3F2FF     jp    FFF2       ldir
```

```
E707  E1            pop   hl
E708  CD9AE7        call  E79A          BASIC-Zeile de suchen
E70B  C5            push  bc
E70C  E5            push  hl
E70D  09            add   hl,bc
E70E  EB            ex    de,hl
E70F  2A89AE        ld    hl,(AE89)     Arrayende
E712  CDCFFF        call  FFCF          hl := hl - de
E715  44            ld    b,h
E716  4D            ld    c,l
E717  EB            ex    de,hl
E718  D1            pop   de
E719  78            ld    a,b
E71A  B1            or    c
E71B  C4F2FF        call  nz,FFF2       ldir
E71E  D1            pop   de
E71F  210000        ld    hl,0000
E722  CDDAFF        call  FFDA          bc := hl - de
E725  C32CF5        jp    F52C          Prg- und Variablenzeiger um bc erhöhen

****************************************** BASIC-Befehl DELETE
E728  CD37E7        call  E737
E72B  CD4ADD        call  DD4A          Ende des Statements, sonst 'Syntax error'
E72E  CD5AE7        call  E75A
E731  CD7AC1        call  C17A
E734  C364C0        jp    C064          zum READY-Modus

*******************************************************
E737  CDB0CE        call  CEB0          Zeilennummernbereich holen
E73A  E5            push  hl
E73B  C5            push  bc
E73C  CDC1E7        call  E7C1          BASIC-Zeile de suchen
E73F  D1            pop   de
E740  E5            push  hl
E741  CDA3E7        call  E7A3          BASIC-Zeile de suchen
E744  223BAE        ld    (AE3B),hl
E747  EB            ex    de,hl
E748  E1            pop   hl
E749  CDCFFF        call  FFCF          hl := hl -de
E74C  223DAE        ld    (AE3D),hl
E74F  3804          jr    c,E755        'Improper argument'
E751  7C            ld    a,h
E752  B5            or    l
E753  E1            pop   hl
E754  C0            ret   nz
E755  1E05          ld    e,05          'Improper argument'
E757  C394CA        jp    CA94          Fehlermeldung ausgeben

*******************************************************
E75A  CD87E6        call  E687          Zeilenadressen durch -Nummern ersetzen
E75D  ED4B3DAE      ld    bc,(AE3D)
E761  2A3BAE        ld    hl,(AE3B)
E764  C30BE7        jp    E70B
```

```
*********************************************** Zeilenadresse holen
E767  23         inc   hl
E768  5E         ld    e,(hl)
E769  23         inc   hl            Nummer oder Adresse nach de
E76A  56         ld    d,(hl)
E76B  23         inc   hl
E76C  FE1D       cp    1D            'Zeilenadresse' ?
E76E  C8         ret   z             ja, fertig
E76F  FE1E       cp    1E            'Zeilennummer' ?
E771  C2EAE8     jp    nz,E8EA       nein, 'Syntax error'
E774  E5         push  hl
E775  CDD6DD     call  DDD6          Zeilennummer nach hl holen
E778  DCB8FF     call  c,FFB8        Vergleich hl <> de
E77B  3009       jr    nc,E786       kleiner, ab Programmstart suchen
E77D  E1         pop   hl
E77E  E5         push  hl
E77F  CDF3E8     call  E8F3          Rest der Zeile überlesen
E782  23         inc   hl            ab Adresse (hl)
E783  CDA7E7     call  E7A7          BASIC-Zeile de suchen
E786  D49AE7     call  nc,E79A       nicht gefunden, ab Programmstart suchen
E789  2B         dec   hl
E78A  EB         ex    de,hl
E78B  E1         pop   hl
E78C  E5         push  hl
E78D  3E1D       ld    a,1D          'Zeilenadresse'
E78F  323AAE     ld    (AE3A),a
E792  2B         dec   hl
E793  72         ld    (hl),d
E794  2B         dec   hl            Zeilenadresse ins Programm
E795  73         ld    (hl),e
E796  2B         dec   hl
E797  77         ld    (hl),a        Token für 'Zeilenadresse'
E798  E1         pop   hl
E799  C9         ret

********************************************* BASIC-Zeile de suchen
E79A  CDA3E7     call  E7A3          BASIC-Zeile de suchen
E79D  D8         ret   c             gefunden ?
E79E  1E08       ld    e,08          'Line does not exist'
E7A0  C394CA     jp    CA94          Fehlermeldung ausgeben

****************************************** BASIC-Zeile in (de) suchen
E7A3  2A81AE     ld    hl,(AE81)     Programmstart
E7A6  23         inc   hl
E7A7  4E         ld    c,(hl)
E7A8  23         inc   hl            Zeilenlänge nach bc
E7A9  46         ld    b,(hl)
E7AA  2B         dec   hl
E7AB  78         ld    a,b
E7AC  B1         or    c             Programmende ?
E7AD  C8         ret   z             nicht gefunden
E7AE  E5         push  hl
E7AF  23         inc   hl
E7B0  23         inc   hl
E7B1  7E         ld    a,(hl)
```

```
E7B2  23            inc   hl          Zeilennummer nach hl
E7B3  66            ld    h,(hl)
E7B4  6F            ld    l,a
E7B5  EB            ex    de,hl
E7B6  CDB8FF        call  FFB8        Vergleich hl <> de
E7B9  EB            ex    de,hl
E7BA  E1            pop   hl
E7BB  3F            ccf
E7BC  D0            ret   nc          größer, nicht gefunden
E7BD  C8            ret   z           gleich, gefunden
E7BE  09            add   hl,bc       Zeilenlänge addieren
E7BF  18E6          jr    E7A7        weitersuchen

****************************************** BASIC-Zeile de suchen
E7C1  2A81AE        ld    hl,(AE81)   Programmstart
E7C4  23            inc   hl
E7C5  E5            push  hl          Zeilenadresse merken
E7C6  4E            ld    c,(hl)
E7C7  23            inc   hl          Zeilenlänge nach bc
E7C8  46            ld    b,(hl)
E7C9  23            inc   hl
E7CA  78            ld    a,b
E7CB  B1            or    c
E7CC  280F          jr    z,E7DD      Programmende ?
E7CE  7E            ld    a,(hl)
E7CF  23            inc   hl          Zeilennummer nach hl
E7D0  66            ld    h,(hl)
E7D1  6F            ld    l,a
E7D2  EB            ex    de,hl
E7D3  CDB8FF        call  FFB8        Vergleich hl <> de
E7D6  EB            ex    de,hl
E7D7  3804          jr    c,E7DD      laufende Zeilennummer größer oder gleich ?
E7D9  E1            pop   hl
E7DA  09            add   hl,bc       Zeilenlänge addieren
E7DB  18E8          jr    E7C5        weitersuchen

E7DD  E1            pop   hl          hl zeigt auf Zeilenadresse
E7DE  C9            ret

****************************************** BASIC-Befehl RENUM
E7DF  110A00        ld    de,000A     10, Default für Startwert
E7E2  2805          jr    z,E7E9
E7E4  FE2C          cp    2C          ','
E7E6  C4E1CE        call  nz,CEE1     Zeilennummer nach de holen
E7E9  D5            push  de
E7EA  110000        ld    de,0000     0
E7ED  CD55DD        call  DD55        folgt Komma ?
E7F0  3005          jr    nc,E7F7     nein
E7F2  FE2C          cp    2C          ','
E7F4  C4E1CE        call  nz,CEE1     Zeilennummer nach de holen
E7F7  D5            push  de
E7F8  110A00        ld    de,000A     10, Default für Inkrement
E7FB  CD55DD        call  DD55        folgt Komma ?
E7FE  DCE1CE        call  c,CEE1      ja, Zeilennummer nach de holen
E801  CD4ADD        call  DD4A        Zeilenende, sonst 'Syntax error'
```

```
E804  E1        pop   hl
E805  EB        ex    de,hl
E806  E3        ex    (sp),hl
E807  EB        ex    de,hl
E808  D5        push  de
E809  E5        push  hl
E80A  CDA3E7    call  E7A3      BASIC-Zeile de suchen
E80D  D1        pop   de
E80E  E5        push  hl
E80F  CDA3E7    call  E7A3      BASIC-Zeile de suchen
E812  EB        ex    de,hl
E813  E1        pop   hl
E814  CDB8FF    call  FFB8      Vergleich hl <> de
E817  DA55E7    jp    c,E755    'Improper argument'
E81A  EB        ex    de,hl
E81B  D1        pop   de
E81C  C1        pop   bc
E81D  D5        push  de
E81E  E5        push  hl
E81F  C5        push  bc
E820  4E        ld    c,(hl)
E821  23        inc   hl
E822  46        ld    b,(hl)
E823  78        ld    a,b
E824  B1        or    c
E825  2813      jr    z,E83A
E827  2B        dec   hl
E828  09        add   hl,bc
E829  7E        ld    a,(hl)
E82A  23        inc   hl
E82B  B6        or    (hl)
E82C  280C      jr    z,E83A
E82E  2B        dec   hl
E82F  C1        pop   bc
E830  E5        push  hl
E831  EB        ex    de,hl
E832  09        add   hl,bc
E833  EB        ex    de,hl
E834  DA55E7    jp    c,E755    'Improper argument'
E837  E1        pop   hl
E838  18E5      jr    E81F

E83A  0164E8    ld    bc,E864
E83D  CDFFE8    call  E8FF
E840  C1        pop   bc
E841  E1        pop   hl
E842  D1        pop   de
E843  C5        push  bc
E844  E5        push  hl
E845  4E        ld    c,(hl)
E846  23        inc   hl
E847  46        ld    b,(hl)
E848  23        inc   hl
E849  78        ld    a,b
E84A  B1        or    c
```

```
E84B  280C      jr    z,E859
E84D  73        ld    (hl),e
E84E  23        inc   hl
E84F  72        ld    (hl),d
E850  23        inc   hl
E851  E1        pop   hl
E852  09        add   hl,bc
E853  C1        pop   bc
E854  EB        ex    de,hl
E855  09        add   hl,bc
E856  EB        ex    de,hl
E857  18EA      jr    E843

E859  E1        pop   hl
E85A  E1        pop   hl
E85B  0188E8    ld    bc,E888
E85E  CDFFE8    call  E8FF
E861  C364C0    jp    C064          zum READY-Modus

E864  CD43E9    call  E943          nächstes Element der Zeile holen
E867  FE02      cp    02
E869  D8        ret   c
E86A  FE1E      cp    1E            'Zeilennummer'
E86C  20F6      jr    nz,E864
E86E  E5        push  hl
E86F  56        ld    d,(hl)
E870  2B        dec   hl
E871  5E        ld    e,(hl)
E872  CDA3E7    call  E7A3          BASIC-Zeile de suchen
E875  300E      jr    nc,E885
E877  2B        dec   hl
E878  EB        ex    de,hl
E879  E1        pop   hl
E87A  E5        push  hl
E87B  72        ld    (hl),d
E87C  2B        dec   hl
E87D  73        ld    (hl),e
E87E  2B        dec   hl
E87F  3E1D      ld    a,1D          'Zeilenadresse'
E881  77        ld    (hl),a
E882  323AAE    ld    (AE3A),a
E885  E1        pop   hl
E886  18DC      jr    E864

E888  CD43E9    call  E943          nächstes Element der Zeile holen
E88B  FE02      cp    02
E88D  D8        ret   c
E88E  FE1E      cp    1E            'Zeilennummer'
E890  20F6      jr    nz,E888
E892  E5        push  hl
E893  56        ld    d,(hl)
E894  2B        dec   hl
E895  5E        ld    e,(hl)
E896  CDD6DD    call  DDD6          Zeilennummer nach hl
E899  CD18CB    call  CB18          'Undefined line in'
```

```
E89C  E1        pop   hl
E89D  18E9      jr    E888

E89F  0601      ld    b,01
E8A1  2B        dec   hl
E8A2  CD43E9    call  E943         nächstes Element der Zeile holen
E8A5  B7        or    a
E8A6  C8        ret   z
E8A7  FE01      cp    01
E8A9  2807      jr    z,E8B2
E8AB  FEA1      cp    A1           'IF'
E8AD  20F3      jr    nz,E8A2
E8AF  04        inc   b
E8B0  18F0      jr    E8A2

E8B2  CD43E9    call  E943         nächstes Element der Zeile holen
E8B5  FE97      cp    97           'ELSE'
E8B7  20EC      jr    nz,E8A5
E8B9  05        dec   b
E8BA  20E6      jr    nz,E8A2
E8BC  CD3FDD    call  DD3F         Blanks überlesen
E8BF  04        inc   b
E8C0  C9        ret

*********************************************** Test auf indizierte Variable
E8C1  7E        ld    a,(hl)
E8C2  FE5B      cp    5B           '['
E8C4  2803      jr    z,E8C9
E8C6  FE28      cp    28           '('
E8C8  C0        ret   nz           kein Index
E8C9  0600      ld    b,00
E8CB  04        inc   b            Klammerverschachtelung erhöhen
E8CC  CD43E9    call  E943         nächstes Element der Zeile holen
E8CF  FE5B      cp    5B           '['
E8D1  28F8      jr    z,E8CB
E8D3  FE28      cp    28           '('
E8D5  28F4      jr    z,E8CB
E8D7  FE5D      cp    5D           ']'
E8D9  280A      jr    z,E8E5
E8DB  FE29      cp    29           ')'
E8DD  2806      jr    z,E8E5
E8DF  FE02      cp    02
E8E1  3807      jr    c,E8EA       'Syntax error'
E8E3  18E7      jr    E8CC

E8E5  05        dec   b            Klammerverschachtelung erniedrigen
E8E6  20E4      jr    nz,E8CC      noch unpaarige Klammern ?
E8E8  23        inc   hl           hl zeigt jetzt hinter Index
E8E9  C9        ret

E8EA  1E02      ld    e,02         'Syntax error'
E8EC  C394CA    jp    CA94         Fehlermeldung ausgeben
```

```
********************************************** BASIC-Befehl DATA
E8EF  0601      ld    b,01       ':', Ende des Statements
E8F1  1802      jr    E8F5

************************************** BASIC-Befehle ELSE, REM und '
E8F3  0600      ld    b,00       0, Ende der Zeile
E8F5  2B        dec   hl
E8F6  CD43E9    call  E943       nächstes Element der Zeile holen
E8F9  B7        or    a
E8FA  C8        ret   z
E8FB  B8        cp    b          Endekennzeichen erreicht ?
E8FC  20F8      jr    nz,E8F6    nein
E8FE  C9        ret

E8FF  CDD2DD    call  DDD2       aktuelle Zeilenadresse nach hl
E902  E5        push  hl
E903  2A81AE    ld    hl,(AE81)  Programmstart
E906  23        inc   hl
E907  7E        ld    a,(hl)
E908  23        inc   hl
E909  B6        or    (hl)
E90A  2813      jr    z,E91F
E90C  23        inc   hl
E90D  CDCEDD    call  DDCE       aktuelle Zeilenadresse setzen
E910  23        inc   hl
E911  C5        push  bc
E912  CDF9FF    call  FFF9       jp (bc)
E915  C1        pop   bc
E916  2B        dec   hl
E917  CD35E9    call  E935
E91A  B7        or    a
E91B  20F4      jr    nz,E911
E91D  18E7      jr    E906

E91F  E1        pop   hl
E920  C3CEDD    jp    DDCE       aktuelle Zeilenadresse setzen

E923  CD35E9    call  E935
E926  B7        or    a
E927  C0        ret   nz
E928  23        inc   hl
E929  7E        ld    a,(hl)
E92A  23        inc   hl
E92B  B6        or    (hl)
E92C  59        ld    e,c
E92D  CA94CA    jp    z,CA94     Fehlermeldung ausgeben

E930  23        inc   hl
E931  54        ld    d,h
E932  5D        ld    e,l
E933  23        inc   hl
E934  C9        ret
```

```
*********************************************************
E935  CD43E9    call   E943       nächstes Element der Zeile holen
E938  FE02      cp     02         Zeilenende ?
E93A  D8        ret    c
E93B  FE97      cp     97         'ELSE'
E93D  C8        ret    z
E93E  FEEB      cp     EB         'THEN'
E940  20F3      jr     nz,E935
E942  C9        ret

************************************** nächstes Element der Zeile holen
E943  CD3FDD    call   DD3F       Blanks überlesen
E946  C8        ret    z
E947  FE0E      cp     0E
E949  381D      jr     c,E968
E94B  FE20      cp     20         ' '
E94D  3829      jr     c,E978
E94F  FE22      cp     22         '"'
E951  2809      jr     z,E95C     String überlesen
E953  FE7C      cp     7C         '|'
E955  2819      jr     z,E970
E957  FEFF      cp     FF         Funktion ?
E959  C0        ret    nz
E95A  23        inc    hl
E95B  C9        ret

E95C  23        inc    hl
E95D  7E        ld     a,(hl)
E95E  FE22      cp     22         '"'
E960  C8        ret    z
E961  B7        or     a
E962  20F8      jr     nz,E95C
E964  2B        dec    hl
E965  3E22      ld     a,22       '"'
E967  C9        ret

E968  FE08      cp     08
E96A  C8        ret    z
E96B  FE07      cp     07
E96D  C8        ret    z
E96E  23        inc    hl
E96F  23        inc    hl
E970  F5        push   af
E971  23        inc    hl
E972  7E        ld     a,(hl)
E973  17        rla
E974  30FB      jr     nc,E971
E976  F1        pop    af
E977  C9        ret

E978  FE18      cp     18         Ziffernkonstante ?
E97A  D8        ret    c
E97B  FE19      cp     19         Ein-Byte-Zahl ?
E97D  2808      jr     z,E987
E97F  FE1F      cp     1F         Fließkommazahl ?
```

```
E981  3803      jr    c,E986      nein, 2-Byte-Zahl
E983  23        inc   hl
E984  23        inc   hl          entsprechende Anzahl
E985  23        inc   hl          Bytes überlesen
E986  23        inc   hl
E987  23        inc   hl
E988  C9        ret

E989  C5        push  bc
E98A  D5        push  de
E98B  E5        push  hl
E98C  0196E9    ld    bc,E996
E98F  CDFFE8    call  E8FF
E992  E1        pop   hl
E993  D1        pop   de
E994  C1        pop   bc
E995  C9        ret

*************************************************************
E996  E5        push  hl
E997  CD43E9    call  E943        nächstes Element der Zeile holen
E99A  D1        pop   de
E99B  FE02      cp    02          Zeilenende ?
E99D  D8        ret   c           ja
E99E  FE0E      cp    0E
E9A0  30F4      jr    nc,E996
E9A2  FE07      cp    07
E9A4  28F0      jr    z,E996
E9A6  FE08      cp    08
E9A8  28EC      jr    z,E996
E9AA  EB        ex    de,hl
E9AB  CD3FDD    call  DD3F        Blanks überlesen
E9AE  FE0D      cp    0D
E9B0  3802      jr    c,E9B4
E9B2  360D      ld    (hl),0D
E9B4  23        inc   hl
E9B5  3600      ld    (hl),00
E9B7  23        inc   hl
E9B8  3600      ld    (hl),00
E9BA  EB        ex    de,hl
E9BB  18D9      jr    E996

**************************************************** BASIC-Befehl RUN
E9BD  CD51DD    call  DD51        Ende des Statements ?
E9C0  EB        ex    de,hl
E9C1  2A81AE    ld    hl,(AE81)   Programmstart als Default
E9C4  EB        ex    de,hl
E9C5  381C      jr    c,E9E3      ja, Ende des Statements
E9C7  FE1E      cp    1E          Zeilennummer ?
E9C9  2815      jr    z,E9E0
E9CB  FE1D      cp    1D          Zeilenadresse ?
E9CD  2811      jr    z,E9E0
E9CF  CD0DEA    call  EA0D
E9D2  2130EA    ld    hl,EA30     Programm von Kassette laden
E9D5  D213BD    jp    nc,BD13     MC BOOT PROGRAM
```

```
E9D8  CDA8EB      call  EBA8        Filetyp testen
E9DB  2A81AE      ld    hl,(AE81)   Programmstart
E9DE  1811        jr    E9F1

E9E0  CD67E7      call  E767        Zeilenadresse holen
E9E3  D5          push  de
E9E4  CDADD2      call  D2AD        Kassetten-I/O abbrechen
E9E7  CD8CC1      call  C18C
E9EA  CD7AC1      call  C17A        CLEAR
E9ED  CD5EC1      call  C15E
E9F0  E1          pop   hl
E9F1  23          inc   hl
E9F2  F1          pop   af
E9F3  C393DD      jp    DD93        zur Interpreterschleife

********************************************* BASIC-Befehl LOAD
E9F6  CD0DEA      call  EA0D
E9F9  3006        jr    nc,EA01
E9FB  CDA8EB      call  EBA8
E9FE  C364C0      jp    C064        zum READY-Modus

EA01  E5          push  hl
EA02  CD01F5      call  F501        auf Platz im Speicher prüfen
EA05  CD30EA      call  EA30        Programm laden
EA08  CA6BCB      jp    z,CB6B
EA0B  E1          pop   hl
EA0C  C9          ret

EA0D  CD8FEB      call  EB8F        Name holen, File öffnen
EA10  E60E        and   0E          Filetyp
EA12  EE02        xor   02
EA14  280B        jr    z,EA21
EA16  CD4ADD      call  DD4A        Ende des Statements, sonst 'Syntax error'
EA19  CD8CC1      call  C18C
EA1C  CD6BC1      call  C16B
EA1F  37          scf
EA20  C9          ret

EA21  CD55DD      call  DD55        folgt Komma ?
EA24  DC91CE      call  c,CE91      ja, 16-Bit-Wert holen
EA27  ED533FAE    ld    (AE3F),de   Startadresse
EA2B  CD4ADD      call  DD4A        Ende des Statements, sonst 'Syntax error'
EA2E  B7          or    a
EA2F  C9          ret

EA30  2A3FAE      ld    hl,(AE3F)   Startadresse
EA33  CD83BC      call  BC83        CAS IN DIRECT
EA36  E5          push  hl
EA37  DC7ABC      call  c,BC7A      CAS IN CLOSE
EA3A  E1          pop   hl
EA3B  C9          ret
```

```
*********************************************** BASIC-Befehl CHAIN
EA3C EEAB      xor   AB          'MERGE'
EA3E 2004      jr    nz,EA44
EA40 CD3FDD    call  DD3F        Blanks überlesen
EA43 37        scf
EA44 9F        sbc   a,a
EA45 3241AE    ld    (AE41),a    Flag für MERGE
EA48 CD8FEB    call  EB8F        Name holen, File öffnen
EA4B 110000    ld    de,0000     Defaultwert Null für Startzeile
EA4E CD55DD    call  DD55        folgt Komma ?
EA51 3006      jr    nc,EA59     nein
EA53 7E        ld    a,(hl)
EA54 FE2C      cp    2C          ','
EA56 C491CE    call  nz,CE91     16-Bit-Wert holen
EA59 D5        push  de          als Startzeile merken
EA5A CD55DD    call  DD55        folgt Komma ?
EA5D 3E00      ld    a,00
EA5F 3009      jr    nc,EA6A     nein
EA61 CD37DD    call  DD37        Test auf nachfolgendes Zeichen
EA64 92        db    92          'DELETE'
EA65 CD37E7    call  E737        Zeilenbereich löschen
EA68 3EFF      ld    a,FF        Flag für DELETE setzen
EA6A F5        push  af
EA6B CD4ADD    call  DD4A        Ende des Statements, sonst 'Syntax error'
EA6E CD1BFB    call  FB1B
EA71 CD3EFC    call  FC3E        Garbage Collection
EA74 CD89E9    call  E989
EA77 CDD2D5    call  D5D2        Funktionen FN löschen
EA7A CD49F5    call  F549
EA7D F1        pop   af
EA7E C5        push  bc
EA7F D5        push  de
EA80 B7        or    a           DELETE ?
EA81 C45AE7    call  nz,E75A     ja, Zeilen löschen
EA84 3A41AE    ld    a,(AE41)    Flag für MERGE
EA87 B7        or    a
EA88 2008      jr    nz,EA92     ja, CHAIN MERGE
EA8A CD6BC1    call  C16B        Variablen löschen
EA8D CDA8EB    call  EBA8        Filetyp prüfen
EA90 1803      jr    EA95

EA92 CD9DEB    call  EB9D        Filetyp testen
EA95 D1        pop   de          Länge der Variablen
EA96 C1        pop   bc          Länge des Stringbereichs
EA97 CD71F5    call  F571        Strings verschieben
EA9A D1        pop   de          Startzeile holen
EA9B 2A81AE    ld    hl,(AE81)   Programmstart als Default
EA9E 7A        ld    a,d
EA9F B3        or    e           keine Startzeile
EAA0 C8        ret   z
EAA1 CD9AE7    call  E79A        BASIC-Zeile de suchen
EAA4 2B        dec   hl
EAA5 C9        ret
```

```
******************************************** BASIC-Befehl MERGE
EAA6 CD8FEB     call  EB8F        Name holen, File öffnen
EAA9 CD4ADD     call  DD4A        Ende des Statements, sonst 'Syntax error'
EAAC CD8CC1     call  C18C        Variablen löschen
EAAF CD9DEB     call  EB9D        Filetyp testen
EAB2 C364C0     jp    C064        zum READY-Modus

*************************************************************
EAB5 CD7AC1     call  C17A
EAB8 CD87E6     call  E687
EABB 2A83AE     ld    hl,(AE83)   Programmende
EABE EB         ex    de,hl
EABF 2A81AE     ld    hl,(AE81)   Programmstart
EAC2 23         inc   hl
EAC3 2283AE     ld    (AE83),hl   Programmende
EAC6 EB         ex    de,hl
EAC7 CDDAFF     call  FFDA        bc := hl - de
EACA EB         ex    de,hl
EACB 2A8DB0     ld    hl,(B08D)   Beginn der Strings
EACE EB         ex    de,hl
EACF 2B         dec   hl
EAD0 CDF5FF     call  FFF5        lddr
EAD3 13         inc   de
EAD4 EB         ex    de,hl
EAD5 E5         push  hl
EAD6 2A83AE     ld    hl,(AE83)   Programmende
EAD9 112000     ld    de,0020
EADC 19         add   hl,de
EADD EB         ex    de,hl
EADE E1         pop   hl
EADF CDB8FF     call  FFB8        Vergleich hl <> de
EAE2 3850       jr    c,EB34      'Memory full'
EAE4 CD84EB     call  EB84
EAE7 B3         or    e
EAE8 2830       jr    z,EB1A
EAEA D5         push  de
EAEB CD84EB     call  EB84
EAEE E5         push  hl
EAEF 7E         ld    a,(hl)
EAF0 23         inc   hl
EAF1 B6         or    (hl)
EAF2 2812       jr    z,EB06
EAF4 23         inc   hl
EAF5 7E         ld    a,(hl)
EAF6 23         inc   hl
EAF7 66         ld    h,(hl)
EAF8 6F         ld    l,a
EAF9 CDB8FF     call  FFB8        Vergleich hl <> de
EAFC E1         pop   hl
EAFD 280F       jr    z,EB0E
EAFF 3006       jr    nc,EB07
EB01 CD48EB     call  EB48
EB04 18E8       jr    EAEE
```

```
EB06  E1          pop   hl
EB07  E3          ex    (sp),hl
EB08  CD5EEB      call  EB5E
EB0B  E1          pop   hl
EB0C  18C7        jr    EAD5

EB0E  E3          ex    (sp),hl
EB0F  CD5EEB      call  EB5E
EB12  E1          pop   hl
EB13  5E          ld    e,(hl)
EB14  23          inc   hl
EB15  56          ld    d,(hl)
EB16  2B          dec   hl
EB17  19          add   hl,de
EB18  18BB        jr    EAD5

EB1A  7E          ld    a,(hl)
EB1B  23          inc   hl
EB1C  B6          or    (hl)
EB1D  2B          dec   hl
EB1E  2805        jr    z,EB25
EB20  CD48EB      call  EB48
EB23  18F5        jr    EB1A

EB25  2A83AE      ld    hl,(AE83)      Programmende
EB28  3600        ld    (hl),00
EB2A  23          inc   hl
EB2B  3600        ld    (hl),00
EB2D  23          inc   hl
EB2E  2283AE      ld    (AE83),hl      Programmende
EB31  C3B1D5      jp    D5B1

EB34  1E07        ld    e,07           'Memory full'
EB36  1802        jr    EB3A

EB38  1E18        ld    e,18           'EOF met'
EB3A  D5          push  de
EB3B  CDADD2      call  D2AD           Kassetten-I/O abbrechen
EB3E  CD8CC1      call  C18C
EB41  CD6BC1      call  C16B
EB44  D1          pop   de
EB45  C394CA      jp    CA94           Fehlermeldung ausgeben

EB48  C5          push  bc
EB49  D5          push  de
EB4A  E5          push  hl
EB4B  4E          ld    c,(hl)
EB4C  23          inc   hl
EB4D  46          ld    b,(hl)
EB4E  2A83AE      ld    hl,(AE83)      Programmende
EB51  EB          ex    de,hl
EB52  E1          pop   hl
EB53  CDF2FF      call  FFF2           ldir
EB56  EB          ex    de,hl
EB57  2283AE      ld    (AE83),hl      Programmende
```

```
EB5A  EB        ex     de,hl
EB5B  D1        pop    de
EB5C  C1        pop    bc
EB5D  C9        ret

EB5E  D5        push   de
EB5F  EB        ex     de,hl
EB60  2A83AE    ld     hl,(AE83)      Programmende
EB63  73        ld     (hl),e
EB64  23        inc    hl
EB65  72        ld     (hl),d
EB66  23        inc    hl
EB67  EB        ex     de,hl
EB68  E3        ex     (sp),hl
EB69  EB        ex     de,hl
EB6A  73        ld     (hl),e
EB6B  23        inc    hl
EB6C  72        ld     (hl),d
EB6D  23        inc    hl
EB6E  D1        pop    de
EB6F  1B        dec    de
EB70  1B        dec    de
EB71  1B        dec    de
EB72  1B        dec    de
EB73  7A        ld     a,d
EB74  B3        or     e
EB75  2809      jr     z,EB80
EB77  CD80BC    call   BC80           CAS IN CHAR
EB7A  30BC      jr     nc,EB38        'EOF met'
EB7C  77        ld     (hl),a
EB7D  23        inc    hl
EB7E  18F2      jr     EB72

EB80  2283AE    ld     (AE83),hl      Programmende
EB83  C9        ret

EB84  CD80BC    call   BC80           CAS IN CHAR
EB87  5F        ld     e,a
EB88  DC80BC    call   c,BC80         CAS IN CHAR
EB8B  30AB      jr     nc,EB38        'EOF met'
EB8D  57        ld     d,a
EB8E  C9        ret

EB8F  CDADD2    call   D2AD           Kassetten-I/O abbrechen
EB92  CD6AD2    call   D26A           Name holen, File öffnen
EB95  3242AE    ld     (AE42),a       Filetyp merken
EB98  ED4343AE  ld     (AE43),bc      Filelänge merken
EB9C  C9        ret

EB9D  3A42AE    ld     a,(AE42)       Filetyp
EBA0  B7        or     a
EBA1  CAB5EA    jp     z,EAB5
EBA4  FE16      cp     16             ASCII-File ?
EBA6  200B      jr     nz,EBB3        'File type error'
EBA8  3A42AE    ld     a,(AE42)       Filetyp
```

```
EBAB FE16       cp    16          ASCII-File ?
EBAD 2840       jr    z,EBEF
EBAF E6FE       and   FE          Bit 0 (geschütztes File) löschen
EBB1 2805       jr    z,EBB8
EBB3 1E19       ld    e,19        'File type error'
EBB5 C394CA     jp    CA94        Fehlermeldung ausgeben

EBB8 CD7AC1     call  C17A
EBBB 2A81AE     ld    hl,(AE81)   Programmstart
EBBE 23         inc   hl
EBBF EB         ex    de,hl
EBC0 2A8DB0     ld    hl,(B08D)   Beginn der Strings
EBC3 0180FF     ld    bc,FF80
EBC6 09         add   hl,bc
EBC7 ED4B43AE   ld    bc,(AE43)   Filelänge
EBCB CDCFFF     call  FFCF        hl := hl - de
EBCE D4BEFF     call  nc,FFBE     Vergleich hl <> bc
EBD1 DA34EB     jp    c,EB34      'Memory full'
EBD4 60         ld    h,b
EBD5 69         ld    l,c
EBD6 19         add   hl,de
EBD7 2283AE     ld    (AE83),hl   Programmende
EBDA 3A42AE     ld    a,(AE42)    Filetyp
EBDD 1F         rra               geschützt ?
EBDE 9F         sbc   a,a
EBDF 3245AE     ld    (AE45),a    Flag für geschütztes Programm setzen
EBE2 EB         ex    de,hl
EBE3 CD83BC     call  BC83        CAS IN DIRECT
EBE6 CA38EB     jp    z,EB38      'EOF met'
EBE9 CDB1D5     call  D5B1
EBEC C398D2     jp    D298        CLOSEIN

EBEF CD7AC1     call  C17A
EBF2 CDCBDD     call  DDCB        aktuelle Zeilenadresse auf Null
EBF5 CD4CCA     call  CA4C        Zeile von Kassette in Eingabepuffer
EBF8 D298D2     jp    nc,D298     CLOSEIN
EBFB CDBCE6     call  E6BC        Zeile in Interpreterkode wandeln
EBFE 38F5       jr    c,EBF5      kein Direkt-Befehl ?
EC00 1E15       ld    e,15        'Direct command found'
EC02 2802       jr    z,EC06
EC04 1E06       ld    e,06        'Overflow'
EC06 C394CA     jp    CA94        Fehlermeldung ausgeben

******************************************* BASIC-Befehl SAVE
EC09 CDADD2     call  D2AD        Kassetten I/O abbrechen
EC0C CD56D2     call  D256        OPENOUT
EC0F 0600       ld    b,00        Filetyp 0, BASIC-Programm
EC11 CD55DD     call  DD55        auf Komma prüfen
EC14 3029       jr    nc,EC3F
EC16 CD37DD     call  DD37        Test auf nachfolgendes Zeichen
EC19 0D         db    0D          'numerische Variable' (A,B,P)
EC1A 23         inc   hl
EC1B 23         inc   hl
EC1C 7E         ld    a,(hl)      Variablenname
EC1D 23         inc   hl
```

```
EC1E  E6DF       and   DF          Klein- in Großbuchstabe wandeln
EC20  F238EC     jp    p,EC38      'Syntax error'
EC23  E5         push  hl
EC24  212CEC     ld    hl,EC2C     Basisadresse der Tabelle
EC27  CD93FF     call  FF93        Tabelle durchsuchen
EC2A  E3         ex    (sp),hl
EC2B  C9         ret

*************************************************************
EC2C  03         db    03          Anzahl der Tabelleneinträge
EC2D  38EC       dw    EC38        Rücksprungadresse wenn nicht gefunden
EC2F  C1         db    C1          'A'
EC30  87EC       dw    EC87
EC32  C2         db    C2          'B'
EC33  5CEC       dw    ECE5
EC35  D0         db    D0          'P'
EC36  3DEC       dw    EC3D

*************************************************************
EC38  1E02       ld    e,02        'Syntax error'
EC3A  C394CA     jp    CA94        Fehlermeldung ausgeben

********************************************************* SAVE ,P
EC3D  0601       ld    b,01        Filetyp 1, protected
EC3F  CD4ADD     call  DD4A        Ende des Statements, sonst 'Syntax error'
EC42  E5         push  hl
EC43  C5         push  bc
EC44  CD87E6     call  E687
EC47  CD89E9     call  E989
EC4A  2A81AE     ld    hl,(AE81)   Programmstart
EC4D  23         inc   hl
EC4E  EB         ex    de,hl
EC4F  2A83AE     ld    hl,(AE83)   Programmende
EC52  CDCFFF     call  FFCF        hl := hl - de
EC55  EB         ex    de,hl
EC56  F1         pop   af
EC57  010000     ld    bc,0000
EC5A  1823       jr    EC7F

********************************************************* SAVE ,B
EC5C  0602       ld    b,02        Filetyp 2, binär
EC5E  CD37DD     call  DD37        Test auf nachfolgendes Zeichen
EC61  2C         db    2C          ','
EC62  CD91CE     call  CE91        16-Bit-Wert holen, Startadresse
EC65  D5         push  de
EC66  CD37DD     call  DD37        Test auf nachfolgendes Zeichen
EC69  2C         db    2C          ','
EC6A  CD91CE     call  CE91        16-Bit-Wert holen, Endadresse
EC6D  D5         push  de
EC6E  CD55DD     call  DD55        folgt Komma ?
EC71  110000     ld    de,0000     Default Null
EC74  DC91CE     call  c,CE91      ja, 16-Bit-Wert holen, Einsprungadresse
EC77  D5         push  de
EC78  CD4ADD     call  DD4A        Ende des Statements, sonst 'Syntax error'
EC7B  78         ld    a,b         Filetyp
```

```
EC7C C1         pop   bc         Einsprungadresse
EC7D D1         pop   de         Endadresse
EC7E E3         ex    (sp),hl    Startadresse
EC7F CD98BC     call  BC98       CAS OUT DIRECT
EC82 D26BCB     jp    nc,CB6B    Unterbrechung durch 'ESC' ?
EC85 1817       jr    EC9E       CLOSEOUT

****************************************************** SAVE ,A
EC87 CD4ADD     call  DD4A       Ende des Statements, sonst 'Syntax error'
EC8A E5         push  hl
EC8B 3E09       ld    a,09       9
EC8D CDA2C1     call  C1A2       Ausgabe auf Kanal 9, Kassette
EC90 F5         push  af
EC91 010100     ld    bc,0001    1
EC94 11FFFF     ld    de,FFFF    bis 65535
EC97 CD0DE1     call  E10D       Zeilen listen
EC9A F1         pop   af
EC9B CDA2C1     call  C1A2       Ausgabe wieder auf Default
EC9E CDA1D2     call  D2A1       CLOSEOUT
ECA1 E1         pop   hl
ECA2 C9         ret

ECA3 CD44ED     call  ED44
ECA6 2005       jr    nz,ECAD
ECA8 CD61DD     call  DD61       Blank, TAB und LF überlesen
ECAB 182F       jr    ECDC

************************************************************
ECAD FE26       cp    26         '&'
ECAF 281C       jr    z,ECCD
ECB1 CD7FFF     call  FF7F       Test auf numerisch
ECB4 3826       jr    c,ECDC
ECB6 CD10FF     call  FF10       Typ auf Integer
ECB9 CDF3FE     call  FEF3       Variable löschen
ECBC 37         scf
ECBD C9         ret

ECBE E5         push  hl
ECBF CDC6EC     call  ECC6
ECC2 D1         pop   de
ECC3 D8         ret   c
ECC4 EB         ex    de,hl
ECC5 C9         ret

ECC6 1600       ld    d,00
ECC8 7E         ld    a,(hl)
ECC9 FE26       cp    26         '&'
ECCB 200F       jr    nz,ECDC
ECCD CD1CEE     call  EE1C
ECD0 EB         ex    de,hl
ECD1 F5         push  af
ECD2 CD0DFF     call  FF0D       Integerzahl hl übernehmen
ECD5 F1         pop   af
ECD6 EB         ex    de,hl
ECD7 D8         ret   c
```

```
ECD8  C8          ret    z
ECD9  C3F3CA      jp     CAF3

ECDC  E5          push   hl
ECDD  7E          ld     a,(hl)
ECDE  23          inc    hl
ECDF  FE2E        cp     2E          '.'
ECE1  CC61DD      call   z,DD61      Blank, TAB und LF überlesen
ECE4  CD83FF      call   FF83        Test auf Ziffer
ECE7  E1          pop    hl
ECE8  3806        jr     c,ECF0
ECEA  7E          ld     a,(hl)
ECEB  EE2E        xor    2E          '.'
ECED  C0          ret    nz
ECEE  23          inc    hl
ECEF  C9          ret

ECF0  CD10FF      call   FF10        Typ auf Integer
ECF3  D5          push   de
ECF4  010000      ld     bc,0000
ECF7  1146AE      ld     de,AE46
ECFA  CD53ED      call   ED53
ECFD  FE2E        cp     2E          '.'
ECFF  200B        jr     nz,ED0C
ED01  CDC9ED      call   EDC9
ED04  CD19FF      call   FF19        Typ auf 'Real'
ED07  0C          inc    c
ED08  CD53ED      call   ED53
ED0B  0D          dec    c
ED0C  F5          push   af
ED0D  3EFF        ld     a,FF
ED0F  12          ld     (de),a
ED10  F1          pop    af
ED11  CD77ED      call   ED77
ED14  D1          pop    de
ED15  5F          ld     e,a
ED16  E5          push   hl
ED17  D5          push   de
ED18  2146AE      ld     hl,AE46
ED1B  CDCEED      call   EDCE
ED1E  D1          pop    de
ED1F  CD27FF      call   FF27        Test auf String
ED22  3008        jr     nc,ED2C
ED24  E5          push   hl
ED25  42          ld     b,d
ED26  CD06FE      call   FE06
ED29  E1          pop    hl
ED2A  3811        jr     c,ED3D
ED2C  7A          ld     a,d
ED2D  4E          ld     c,(hl)
ED2E  23          inc    hl
ED2F  CD94BD      call   BD94        4-Byte-Integer*256 nach Fließkomma
ED32  7B          ld     a,e
ED33  CD55BD      call   BD55        Zahl mit 10↑a multiplizieren
ED36  EB          ex     de,hl
```

```
ED37  CD16FF    call   FF16       Variablentyp auf Fließkomma setzen
ED3A  DC3DBD    call   c,BD3D     Variable von (de) nach (hl)
ED3D  3E0A      ld     a,0A
ED3F  E1        pop    hl
ED40  D8        ret    c
ED41  C3F3CA    jp     CAF3

ED44  CD61DD    call   DD61       Blank, TAB und LF überlesen
ED47  23        inc    hl
ED48  16FF      ld     d,FF
ED4A  FE2D      cp     2D         '-'
ED4C  C8        ret    z
ED4D  14        inc    d
ED4E  FE2B      cp     2B         '+'
ED50  C8        ret    z
ED51  2B        dec    hl
ED52  C9        ret

ED53  E5        push   hl
ED54  CD61DD    call   DD61       Blank, TAB und LF überlesen
ED57  23        inc    hl
ED58  CD83FF    call   FF83       Test auf Ziffer
ED5B  3804      jr     c,ED61
ED5D  E1        pop    hl
ED5E  C38AFF    jp     FF8A       Klein- in Großbuchstaben wandeln

ED61  E3        ex     (sp),hl
ED62  E1        pop    hl
ED63  D630      sub    30         '0'
ED65  12        ld     (de),a
ED66  B0        or     b
ED67  2807      jr     z,ED70
ED69  78        ld     a,b
ED6A  04        inc    b
ED6B  FE0C      cp     0C
ED6D  3001      jr     nc,ED70
ED6F  13        inc    de
ED70  79        ld     a,c
ED71  B7        or     a
ED72  28DF      jr     z,ED53
ED74  0C        inc    c
ED75  18DC      jr     ED53

ED77  FE45      cp     45         'E'
ED79  2010      jr     nz,ED8B
ED7B  E5        push   hl
ED7C  CDC9ED    call   EDC9
ED7F  CD44ED    call   ED44
ED82  CC61DD    call   z,DD61     Blank, TAB und LF überlesen
ED85  CD83FF    call   FF83       Test auf Ziffer
ED88  3804      jr     c,ED8E
ED8A  E1        pop    hl
ED8B  AF        xor    a
ED8C  181E      jr     EDAC
```

```
ED8E E3        ex    (sp),hl
ED8F E1        pop   hl
ED90 CD19FF    call  FF19        Typ auf 'Real' setzen
ED93 D5        push  de
ED94 C5        push  bc
ED95 CD35EE    call  EE35
ED98 3009      jr    nc,EDA3
ED9A 7B        ld    a,e
ED9B D664      sub   64          100
ED9D 7A        ld    a,d
ED9E DE00      sbc   a,00
EDA0 7B        ld    a,e
EDA1 3802      jr    c,EDA5
EDA3 3E7F      ld    a,7F
EDA5 C1        pop   bc
EDA6 D1        pop   de
EDA7 14        inc   d
EDA8 2002      jr    nz,EDAC
EDAA 2F        cpl   a
EDAB 3C        inc   a
EDAC C680      add   a,80
EDAE 5F        ld    e,a
EDAF 78        ld    a,b
EDB0 D60C      sub   0C
EDB2 3001      jr    nc,EDB5
EDB4 AF        xor   a
EDB5 91        sub   c
EDB6 3009      jr    nc,EDC1
EDB8 83        add   a,e
EDB9 3801      jr    c,EDBC
EDBB AF        xor   a
EDBC FE01      cp    01
EDBE CE80      adc   a,80
EDC0 C9        ret

EDC1 83        add   a,e
EDC2 3002      jr    nc,EDC6
EDC4 3EFF      ld    a,FF
EDC6 D680      sub   80
EDC8 C9        ret

EDC9 CD61DD    call  DD61        Blank, TAB und LF überlesen
EDCC 23        inc   hl
EDCD C9        ret

EDCE EB        ex    de,hl
EDCF 2158AE    ld    hl,AE58
EDD2 010105    ld    bc,0501
EDD5 2B        dec   hl
EDD6 3600      ld    (hl),00
EDD8 10FB      djnz  EDD5
EDDA 1A        ld    a,(de)
EDDB FEFF      cp    FF
EDDD C8        ret   z
EDDE 77        ld    (hl),a
```

```
EDDF 2153AE     ld    hl,AE53
EDE2 13         inc   de
EDE3 1A         ld    a,(de)
EDE4 FEFF       cp    FF
EDE6 C8         ret   z
EDE7 D5         push  de
EDE8 41         ld    b,c
EDE9 1600       ld    d,00
EDEB E5         push  hl
EDEC 5E         ld    e,(hl)
EDED 62         ld    h,d
EDEE 6B         ld    l,e
EDEF 29         add   hl,hl
EDF0 29         add   hl,hl
EDF1 19         add   hl,de          mal 10
EDF2 29         add   hl,hl
EDF3 5F         ld    e,a            plus nächste Ziffer
EDF4 19         add   hl,de
EDF5 5D         ld    e,l
EDF6 7C         ld    a,h
EDF7 E1         pop   hl
EDF8 73         ld    (hl),e
EDF9 23         inc   hl
EDFA 10EF       djnz  EDEB
EDFC D1         pop   de
EDFD B7         or    a
EDFE 28DF       jr    z,EDDF
EE00 77         ld    (hl),a
EE01 0C         inc   c
EE02 18DB       jr    EDDF

EE04 C5         push  bc
EE05 E5         push  hl
EE06 CD35EE     call  EE35
EE09 EB         ex    de,hl
EE0A CD0DFF     call  FF0D           Integerzahl hl übernehmen
EE0D EB         ex    de,hl
EE0E C1         pop   bc
EE0F 3006       jr    nc,EE17
EE11 7A         ld    a,d
EE12 B3         or    e
EE13 C6FF       add   a,FF
EE15 3803       jr    c,EE1A
EE17 50         ld    d,b
EE18 59         ld    e,c
EE19 EB         ex    de,hl
EE1A C1         pop   bc
EE1B C9         ret

EE1C 23         inc   hl
EE1D CD61DD     call  DD61           Blank, TAB und LF überlesen
EE20 CD8AFF     call  FF8A           Klein- in Großbuchstaben wandeln
EE23 0602       ld    b,02           Basis 2, binär
EE25 FE58       cp    58             'X'
EE27 2806       jr    z,EE2F
```

```
EE29  0610      ld    b,10       Basis 16, hex
EE2B  FE48      cp    48         'H'
EE2D  2004      jr    nz,EE33
EE2F  23        inc   hl
EE30  CD61DD    call  DD61       Blank, TAB und LF überlesen
EE33  1802      jr    EE37

EE35  060A      ld    b,0A       Basis 10, dezimal
EE37  EB        ex    de,hl
EE38  CD61EE    call  EE61       (Hex-) Ziffer nach binär wandeln
EE3B  2600      ld    h,00
EE3D  6F        ld    l,a
EE3E  301E      jr    nc,EE5E
EE40  0E00      ld    c,00
EE42  CD61EE    call  EE61       (Hex-) Ziffer nach binär wandeln
EE45  3014      jr    nc,EE5B
EE47  D5        push  de
EE48  1600      ld    d,00
EE4A  5F        ld    e,a
EE4B  D5        push  de
EE4C  58        ld    e,b        Basis des Zahlensystems
EE4D  CDBEBD    call  BDBE       Integermultiplikation ohne Vorzeichen
EE50  D1        pop   de
EE51  3803      jr    c,EE56
EE53  19        add   hl,de
EE54  3002      jr    nc,EE58
EE56  0EFF      ld    c,FF
EE58  D1        pop   de
EE59  18E7      jr    EE42

EE5B  79        ld    a,c
EE5C  FE01      cp    01
EE5E  EB        ex    de,hl
EE5F  78        ld    a,b
EE60  C9        ret

**************************************** (Hex-) Ziffer nach binär wandeln
EE61  1A        ld    a,(de)     Zeichen holen
EE62  13        inc   de
EE63  CD83FF    call  FF83       Test auf Ziffer
EE66  380A      jr    c,EE72     ja
EE68  CD8AFF    call  FF8A       Klein- in Großbuchstaben wandeln
EE6B  FE41      cp    41'        A'
EE6D  3F        ccf
EE6E  3005      jr    nc,EE75    kleiner 'A', Fehler
EE70  D607      sub   07'        A'-('9'+1)
EE72  D630      sub   30'        0'
EE74  B8        cp    b
EE75  D8        ret   c          kein Fehler ?
EE76  1B        dec   de
EE77  AF        xor   a          Carry löschen
EE78  C9        ret
```

```
******************************************* Zeilennummer ausgeben
EE79  CD0DFF       call  FF0D        Integerzahl in hl übernehmen
EE7C  CD82EE       call  EE82        in ASCII-Darstellung umwandeln
EE7F  C341C3       jp    C341        String ausgeben

***************************************** Integerzahl nach ASCII wandeln
EE82  D5           push  de
EE83  C5           push  bc
EE84  CDC3FC       call  FCC3
EE87  AF           xor   a
EE88  CDA7EE       call  EEA7
EE8B  23           inc   hl
EE8C  C1           pop   bc
EE8D  D1           pop   de
EE8E  C9           ret

*****************************************************
EE8F  D5           push  de
EE90  C5           push  bc
EE91  AF           xor   a           Null
EE92  CD9FEE       call  EE9F        Zahl in formatierten String wandeln
EE95  C1           pop   bc
EE96  D1           pop   de
EE97  7E           ld    a,(hl)
EE98  FE20         cp    20          ' '
EE9A  C0           ret   nz
EE9B  23           inc   hl
EE9C  C9           ret

*****************************************************
EE9D  3E40         ld    a,40
EE9F  226EAE       ld    (AE6E),hl   Zahl in formatierten String wandeln
EEA2  F5           push  af
EEA3  CDB3FC       call  FCB3
EEA6  F1           pop   af
EEA7  C5           push  bc
EEA8  57           ld    d,a
EEA9  D5           push  de
EEAA  EB           ex    de,hl
EEAB  2168AE       ld    hl,AE68
EEAE  3600         ld    (hl),00
EEB0  2270AE       ld    (AE70),hl
EEB3  CDB7F0       call  F0B7
EEB6  D1           pop   de
EEB7  CDD4EE       call  EED4
EEBA  CD3DF0       call  F03D
EEBD  58           ld    e,b
EEBE  C1           pop   bc
EEBF  7B           ld    a,e
EEC0  B7           or    a
EEC1  CC50F0       call  z,F050
EEC4  CD5FF0       call  F05F
EEC7  CD69F0       call  F069
EECA  CD7CF0       call  F07C
EECD  7A           ld    a,d
```

```
EECE  1F        rra
EECF  D0        ret     nc
EED0  2B        dec     hl              Überlauf,
EED1  3625      ld      (hl),25         '%' vor formatierte Zahl
EED3  C9        ret

EED4  7A        ld      a,d
EED5  87        add     a,a
EED6  3029      jr      nc,EF01
EED8  FA27EF    jp      m,EF27
EEDB  7B        ld      a,e
EEDC  81        add     a,c
EEDD  D60A      sub     0A              10
EEDF  FA88EF    jp      m,EF88
EEE2  1601      ld      d,01
EEE4  41        ld      b,c
EEE5  79        ld      a,c
EEE6  B7        or      a
EEE7  2815      jr      z,EEFE
EEE9  83        add     a,e
EEEA  3D        dec     a
EEEB  5F        ld      e,a
EEEC  CD0EF0    call    F00E
EEEF  0601      ld      b,01
EEF1  79        ld      a,c
EEF2  FE07      cp      07
EEF4  3804      jr      c,EEFA
EEF6  CB72      bit     6,d
EEF8  2026      jr      nz,EF20
EEFA  B8        cp      b
EEFB  C4A0EF    call    nz,EFA0
EEFE  C362EF    jp      EF62

EF01  7B        ld      a,e
EF02  B7        or      a
EF03  FA0AEF    jp      m,EF0A
EF06  20DC      jr      nz,EEE4
EF08  41        ld      b,c
EF09  C9        ret

EF0A  43        ld      b,e
EF0B  CD0EF0    call    F00E
EF0E  78        ld      a,b
EF0F  B7        or      a
EF10  28F6      jr      z,EF08
EF12  93        sub     e
EF13  58        ld      e,b
EF14  47        ld      b,a
EF15  81        add     a,c
EF16  83        add     a,e
EF17  FAE4EE    jp      m,EEE4
EF1A  CDB4EF    call    EFB4
EF1D  C3A0EF    jp      EFA0
```

```
EF20  3E06      ld    a,06
EF22  326EAE    ld    (AE6E),a
EF25  1824      jr    EF4B

EF27  0680      ld    b,80
EF29  CD25F0    call  F025
EF2C  3004      jr    nc,EF32
EF2E  CD96F0    call  F096
EF31  AF        xor   a
EF32  47        ld    b,a
EF33  CC36F0    call  z,F036
EF36  200C      jr    nz,EF44
EF38  04        inc   b
EF39  3A6EAE    ld    a,(AE6E)
EF3C  B7        or    a
EF3D  2805      jr    z,EF44
EF3F  05        dec   b
EF40  3C        inc   a
EF41  326EAE    ld    (AE6E),a
EF44  79        ld    a,c
EF45  B7        or    a
EF46  2804      jr    z,EF4C
EF48  83        add   a,e
EF49  90        sub   b
EF4A  5F        ld    e,a
EF4B  78        ld    a,b
EF4C  F5        push  af
EF4D  47        ld    b,a
EF4E  CD8BEF    call  EF8B
EF51  F1        pop   af
EF52  B8        cp    b
EF53  280D      jr    z,EF62
EF55  1C        inc   e
EF56  23        inc   hl
EF57  05        dec   b
EF58  E5        push  hl
EF59  7E        ld    a,(hl)
EF5A  FE2E      cp    2E          '.'
EF5C  2001      jr    nz,EF5F
EF5E  23        inc   hl
EF5F  3631      ld    (hl),31     '1'
EF61  E1        pop   hl
EF62  3E45      ld    a,45        'E'
EF64  CD6FF0    call  F06F
EF67  7B        ld    a,e
EF68  87        add   a,a
EF69  3E2B      ld    a,2B        '+'
EF6B  3005      jr    nc,EF72
EF6D  AF        xor   a
EF6E  93        sub   e
EF6F  5F        ld    e,a
EF70  3E2D      ld    a,2D        '-'
EF72  CD6FF0    call  F06F
EF75  7B        ld    a,e
EF76  0E2F      ld    c,2F        '0'-1
```

```
EF78  0C        inc   c
EF79  D60A      sub   0A          10
EF7B  30FB      jr    nc,EF78
EF7D  5F        ld    e,a
EF7E  79        ld    a,c
EF7F  CD6FF0    call  F06F
EF82  7B        ld    a,e
EF83  C63A      add   a,3A        '9'+1
EF85  C36FF0    jp    F06F

EF88  CDB4EF    call  EFB4
EF8B  CD36F0    call  F036
EF8E  80        add   a,b
EF8F  B9        cp    c
EF90  3005      jr    nc,EF97
EF92  CDC8EF    call  EFC8
EF95  1804      jr    EF9B

EF97  91        sub   c
EF98  C4EFEF    call  nz,EFEF
EF9B  3A6EAE    ld    a,AE6E
EF9E  B7        or    a
EF9F  C8        ret   z
EFA0  0E2E      ld    c,2E        '.'
EFA2  78        ld    a,b
EFA3  C5        push  bc
EFA4  47        ld    b,a
EFA5  04        inc   b
EFA6  85        add   a,l
EFA7  6F        ld    l,a
EFA8  8C        adc   a,h
EFA9  95        sub   l
EFAA  67        ld    h,a
EFAB  2B        dec   hl
EFAC  79        ld    a,c
EFAD  4E        ld    c,(hl)
EFAE  77        ld    (hl),a
EFAF  05        dec   b
EFB0  20F9      jr    nz,EFAB
EFB2  C1        pop   bc
EFB3  C9        ret

EFB4  7B        ld    a,e
EFB5  81        add   a,c
EFB6  47        ld    b,a
EFB7  F0        ret   p
EFB8  2F        cpl   a
EFB9  3C        inc   a
EFBA  0614      ld    b,14
EFBC  B8        cp    b
EFBD  3001      jr    nc,EFC0
EFBF  47        ld    b,a
EFC0  2B        dec   hl
EFC1  3630      ld    (hl),30     '0'
EFC3  0C        inc   c
```

```
EFC4  05           dec    b
EFC5  20F9         jr     nz,EFC0
EFC7  C9           ret

EFC8  E5           push   hl
EFC9  4F           ld     c,a
EFCA  85           add    a,l
EFCB  6F           ld     l,a
EFCC  8C           adc    a,h
EFCD  95           sub    l
EFCE  67           ld     h,a
EFCF  7E           ld     a,(hl)
EFD0  3600         ld     (hl),00
EFD2  2270AE       ld     (AE70),hl
EFD5  FE35         cp     35              '5'
EFD7  D4E1EF       call   nc,EFE1
EFDA  E1           pop    hl
EFDB  D8           ret    c
EFDC  2B           dec    hl
EFDD  3631         ld     (hl),31         '1'
EFDF  04           inc    b
EFE0  C9           ret

EFE1  79           ld     a,c
EFE2  B7           or     a
EFE3  C8           ret    z
EFE4  2B           dec    hl
EFE5  0D           dec    c
EFE6  7E           ld     a,(hl)
EFE7  34           inc    (hl)
EFE8  FE39         cp     39              '9'
EFEA  D8           ret    c
EFEB  3630         ld     (hl),30         '0'
EFED  18F2         jr     EFE1

EFEF  D5           push   de
EFF0  C5           push   bc
EFF1  EB           ex     de,hl
EFF2  47           ld     b,a
EFF3  7B           ld     a,e
EFF4  90           sub    b
EFF5  6F           ld     l,a
EFF6  9F           sbc    a,a
EFF7  82           add    a,d
EFF8  67           ld     h,a
EFF9  E5           push   hl
EFFA  0C           inc    c
EFFB  1804         jr     F001

EFFD  1A           ld     a,(de)
EFFE  13           inc    de
EFFF  77           ld     (hl),a
F000  23           inc    hl
F001  0D           dec    c
F002  20F9         jr     nz,EFFD
```

```
F004  3630      ld    (hl),30      '0'
F006  23        inc   hl
F007  05        dec   b
F008  20FA      jr    nz,F004
F00A  E1        pop   hl
F00B  C1        pop   bc
F00C  D1        pop   de
F00D  C9        ret

F00E  E5        push  hl
F00F  2A70AE    ld    hl,(AE70)
F012  2B        dec   hl
F013  7E        ld    a,(hl)
F014  23        inc   hl
F015  FE30      cp    30           '0'
F017  2005      jr    nz,F01E
F019  2B        dec   hl
F01A  0D        dec   c
F01B  04        inc   b
F01C  20F4      jr    nz,F012
F01E  3600      ld    (hl),00
F020  2270AE    ld    (AE70),hl
F023  E1        pop   hl
F024  C9        ret

F025  CD9BF0    call  F09B
F028  9F        sbc   a,a
F029  3C        inc   a
F02A  47        ld    b,a
F02B  7A        ld    a,d
F02C  E604      and   04
F02E  2801      jr    z,F031
F030  04        inc   b
F031  3A6FAE    ld    a,(AE6F)
F034  90        sub   b
F035  C9        ret

F036  3A6EAE    ld    a,(AE6E)
F039  B7        or    a
F03A  C8        ret   z
F03B  3D        dec   a
F03C  C9        ret

F03D  7A        ld    a,d
F03E  E602      and   02
F040  C8        ret   z
F041  78        ld    a,b
F042  D603      sub   03
F044  D8        ret   c
F045  C8        ret   z
F046  F5        push  af
F047  0E2C      ld    c,2C         ','
F049  CDA3EF    call  EFA3
F04C  04        inc   b
F04D  F1        pop   af
```

```
F04E  18F2      jr    F042

F050  7A        ld    a,d
F051  87        add   a,a
F052  3007      jr    nc,F05B
F054  C5        push  bc
F055  CD25F0    call  F025
F058  C1        pop   bc
F059  D8        ret   c
F05A  C8        ret   z
F05B  3E30      ld    a,30         '0'
F05D  1806      jr    F065

F05F  7A        ld    a,d
F060  E604      and   04
F062  C8        ret   z
F063  3E24      ld    a,24         '$'
F065  1C        inc   e
F066  2B        dec   hl
F067  77        ld    (hl),a
F068  C9        ret

F069  CD9BF0    call  F09B
F06C  C8        ret   z
F06D  30F6      jr    nc,F065
F06F  E5        push  hl
F070  2A70AE    ld    hl,(AE70)
F073  77        ld    (hl),a
F074  23        inc   hl
F075  3600      ld    (hl),00
F077  2270AE    ld    (AE70),hl
F07A  E1        pop   hl
F07B  C9        ret

F07C  7A        ld    a,d
F07D  B7        or    a
F07E  F0        ret   p
F07F  3A6FAE    ld    a,(AE6F)
F082  93        sub   e
F083  C8        ret   z
F084  3810      jr    c,F096
F086  47        ld    b,a
F087  7A        ld    a,d
F088  E620      and   20
F08A  3E2A      ld    a,2A         '*'
F08C  2002      jr    nz,F090
F08E  3E20      ld    a,20         ' '
F090  2B        dec   hl
F091  77        ld    (hl),a
F092  05        dec   b
F093  20FB      jr    nz,F090
F095  C9        ret
```

```
F096  7A        ld    a,d
F097  F601      or    01
F099  57        ld    d,a
F09A  C9        ret

F09B  78        ld    a,b
F09C  062D      ld    b,2D        '-'
F09E  87        add   a,a
F09F  380F      jr    c,F0B0
F0A1  7A        ld    a,d
F0A2  E698      and   98
F0A4  EE80      xor   80
F0A6  37        scf
F0A7  C8        ret   z
F0A8  062B      ld    b,2B        '+'
F0AA  E608      and   08
F0AC  2002      jr    nz,F0B0
F0AE  0620      ld    b,20        ' '
F0B0  7A        ld    a,d
F0B1  F6EF      or    EF
F0B3  C610      add   a,10
F0B5  78        ld    a,b
F0B6  C9        ret

F0B7  E5        push  hl
F0B8  EB        ex    de,hl
F0B9  CDDDF0    call  F0DD
F0BC  E1        pop   hl
F0BD  78        ld    a,b
F0BE  87        add   a,a
F0BF  4F        ld    c,a
F0C0  C8        ret   z
F0C1  1A        ld    a,(de)      Byte laden
F0C2  E60F      and   0F          unteres Nibble isolieren
F0C4  C630      add   a,30        '0', nach ASCII
F0C6  2B        dec   hl
F0C7  77        ld    (hl),a      in Puffer
F0C8  1A        ld    a,(de)      Byte laden
F0C9  E6F0      and   F0          oberes Nibble isolieren
F0CB  1F        rra
F0CC  1F        rra               nach unten schieben
F0CD  1F        rra
F0CE  1F        rra
F0CF  C630      add   a,30        '0', nach ASCII
F0D1  2B        dec   hl          davor
F0D2  77        ld    (hl),a      in Puffer
F0D3  13        inc   de
F0D4  05        dec   b
F0D5  20EA      jr    nz,F0C1
F0D7  FE30      cp    30          '0'
F0D9  C0        ret   nz
F0DA  0D        dec   c
F0DB  23        inc   hl
F0DC  C9        ret
```

```
F0DD  1146AE        ld     de,AE46
F0E0  AF            xor    a
F0E1  47            ld     b,a
F0E2  B6            or     (hl)
F0E3  2B            dec    hl
F0E4  2004          jr     nz,F0EA
F0E6  0D            dec    c
F0E7  20F9          jr     nz,F0E2
F0E9  C9            ret

F0EA  37            scf
F0EB  8F            adc    a,a
F0EC  30FD          jr     nc,F0EB
F0EE  EB            ex     de,hl
F0EF  D5            push   de
F0F0  57            ld     d,a
F0F1  1811          jr     F104

F0F3  1A            ld     a,(de)
F0F4  1B            dec    de
F0F5  D5            push   de
F0F6  37            scf
F0F7  8F            adc    a,a
F0F8  57            ld     d,a
F0F9  58            ld     e,b
F0FA  7E            ld     a,(hl)
F0FB  8F            adc    a,a
F0FC  27            daa
F0FD  77            ld     (hl),a
F0FE  23            inc    hl
F0FF  1D            dec    e
F100  20F8          jr     nz,F0FA
F102  3003          jr     nc,F107
F104  04            inc    b
F105  3601          ld     (hl),01
F107  2146AE        ld     hl,AE46
F10A  7A            ld     a,d
F10B  87            add    a,a
F10C  20EA          jr     nz,F0F8
F10E  D1            pop    de
F10F  0D            dec    c
F110  20E1          jr     nz,F0F3
F112  EB            ex     de,hl
F113  C9            ret

********************************************* Umwandlung nach Binär
F114  110101        ld     de,0101
F117  1803          jr     F11C

********************************************* Umwandlung nach Hex
F119  110F04        ld     de,040F
F11C  D5            push   de
F11D  79            ld     a,c
F11E  CD4BFF        call   FF4B          Variablentyp setzen
F121  E3            ex     (sp),hl
```

```
F122  E5            push  hl
F123  C5            push  bc
F124  CDC2FE        call  FEC2        UNT
F127  1157AE        ld    de,AE57
F12A  AF            xor   a
F12B  12            ld    (de),a
F12C  F1            pop   af
F12D  C1            pop   bc
F12E  D601          sub   01
F130  CE00          adc   a,00
F132  F5            push  af
F133  7D            ld    a,l
F134  A1            and   c
F135  F6F0          or    F0
F137  27            daa
F138  C6A0          add   a,A0
F13A  CE40          adc   a,40
F13C  1B            dec   de
F13D  12            ld    (de),a
F13E  7D            ld    a,l
F13F  B1            or    c
F140  A9            xor   c
F141  6F            ld    l,a
F142  B4            or    h
F143  280E          jr    z,F153
F145  C5            push  bc
F146  7C            ld    a,h
F147  1F            rra
F148  67            ld    h,a
F149  7D            ld    a,l
F14A  1F            rra
F14B  6F            ld    l,a
F14C  05            dec   b
F14D  20F7          jr    nz,F146
F14F  C1            pop   bc
F150  F1            pop   af
F151  18DB          jr    F12E

F153  F1            pop   af
F154  20D8          jr    nz,F12E
F156  E1            pop   hl
F157  C9            ret

****************************************** BASIC-Funktion PEEK
F158  CDC2FE        call  FEC2        UNT
F15B  E7            rst   4           READ RAM; ld a,(hl)
F15C  C30AFF        jp    FF0A        Akkuinhalt als Integerzahl übernehmen

****************************************** BASIC-Befehl POKE
F15F  CD91CE        call  CE91        16-Bit-Adresse holen
F162  D5            push  de
F163  CD37DD        call  DD37        Test auf nachfolgendes Zeichen
F166  2C            db    2C          ','
F167  CD67CE        call  CE67        8-Bit-Wert holen
F16A  D1            pop   de
```

```
F16B  12          ld    (de),a       Wert in Adresse schreiben
F16C  C9          ret

*************************************************** BASIC-Funktion INP
F16D  CD8DFE      call  FE8D         CINT
F170  44          ld    b,h          Portadresse nach bc
F171  4D          ld    c,l
F172  ED78        in    a,(c)        Port lesen
F174  C30AFF      jp    FF0A         Akkuinhalt als Integerzahl übernehmen

*************************************************** BASIC-Befehl OUT
F177  CD94F1      call  F194         Adresse und Wert holen
F17A  ED79        out   (c),a        ausgeben
F17C  C9          ret

*************************************************** BASIC-Befehl WAIT
F17D  CD94F1      call  F194         16- und 8-Bit-Wert holen
F180  57          ld    d,a          8-Bit-Wert nach d
F181  1E00        ld    e,00         3. Parameter Null
F183  2808        jr    z,F18D       kein weiterer Wert mehr ?
F185  CD37DD      call  DD37         Test auf nachfolgendes Zeichen
F188  2C          db    2C           ','
F189  CD67CE      call  CE67         8-Bit-Wert holen
F18C  5F          ld    e,a          und nach e
F18D  ED78        in    a,(c)        Port lesen
F18F  AB          xor   e
F190  A2          and   d            verknüpfen
F191  28FA        jr    z,F18D       und warten
F193  C9          ret

*************************************************** 16-Bit und 8-Bit-Wert holen
F194  CD91CE      call  CE91         16-Bit-Wert holen
F197  42          ld    b,d
F198  4B          ld    c,e          nach bc
F199  CD37DD      call  DD37         Test auf nachfolgendes Zeichen
F19C  2C          db    2C           ','
F19D  C367CE      jp    CE67         und 8-Bit-Wert holen

*************************************************** Befehlserweiterung
F1A0  23          inc   hl           Programmzeiger erhöhen
F1A1  7E          ld    a,(hl)
F1A2  B7          or    a            folgt Nullbyte ?
F1A3  2010        jr    nz,F1B5      nein, 'Unknown command'
F1A5  23          inc   hl
F1A6  E5          push  hl
F1A7  CDD4BC      call  BCD4         KL FIND COMMAND
F1AA  EB          ex    de,hl        Befehlsadresse nach de
F1AB  E1          pop   hl
F1AC  3007        jr    nc,F1B5      nicht gefunden, 'Unknown command'
F1AE  7E          ld    a,(hl)       Zeichen holen
F1AF  23          inc   hl           Befehlswort überlesen
F1B0  17          rla                Bit 7 gesetzt ?
F1B1  30FB        jr    nc,F1AE      nein
F1B3  180A        jr    F1BF         zum CALL-Befehl
```

```
F1B5  1E1C        ld    e,1C          'Unknown command'
F1B7  C394CA      jp    CA94          Fehlermeldung ausgeben

****************************************** BASIC-Befehl CALL
F1BA  CD91CE      call  CE91          Adresse holen
F1BD  0EFF        ld    c,FF          FF = Ram selektiert
F1BF  ED5372AE    ld    (AE72),de     Adresse nach AE72
F1C3  79          ld    a,c
F1C4  3274AE      ld    (AE74),a      Konfigurationsbyte nach AE74
F1C7  ED7377AE    ld    (AE77),sp     Stackpointer retten
F1CB  0620        ld    b,20          maximal 32 Parameter
F1CD  CD55DD      call  DD55          folgt Komma?
F1D0  3006        jr    nc,F1D8       nein
F1D2  CD91CE      call  CE91          Parameter holen
F1D5  D5          push  de            und auf den Stack
F1D6  10F5        djnz  F1CD          nächsten Parameter holen
F1D8  CD4ADD      call  DD4A          Ende des Statements, sonst 'Syntax error'
F1DB  2275AE      ld    (AE75),hl     hl-Register retten
F1DE  3E20        ld    a,20
F1E0  90          sub   b             Anzahl der Parameter in Akku
F1E1  DD210000    ld    ix,0000
F1E5  DD39        add   ix,sp         Stackpointer nach ix
F1E7  DF          rst   3             Routine ausführen
F1E8  72AE        dw    AE72
F1EA  ED7B77AE    ld    sp,(AE77)     Stackpointer zurückholen
F1EE  2A75AE      ld    hl,(AE75)     hl-Register zurück
F1F1  C9          ret

***************************************** TAB-Stops initialisieren
F1F2  3E0D        ld    a,0D          13
F1F4  1803        jr    F1F9

****************************************** BASIC-Befehl ZONE
F1F6  CD6DCE      call  CE6D          8-Bit-Wert ungleich Null holen
F1F9  3279AE      ld    (AE79),a      Tabulatorweite merken
F1FC  C9          ret

***************************************** BASIC-Befehl PRINT
F1FD  CDC6C1      call  C1C6          Kanalnummer
F200  F5          push  af
F201  CD08F2      call  F208          PRINT-Ausgabe
F204  F1          pop   af
F205  C3A2C1      jp    C1A2          Kanalnummer zurücksetzen

F208  CD51DD      call  DD51          Ende des Statements ?
F20B  DA4EC3      jp    c,C34E        ja, LF ausgeben
F20E  FEED        cp    ED            'USING'
F210  CAC4F2      jp    z,F2C4
F213  EB          ex    de,hl
F214  2124F2      ld    hl,F224       Basisadresse der Tabelle
F217  CD93FF      call  FF93          Tabelle durchsuchen
F21A  EB          ex    de,hl
F21B  CDFBFF      call  FFFB          jp (de)
F21E  CD51DD      call  DD51          Ende des Statements ?
F221  30EB        jr    nc,F20E       nein, weitermachen
```

```
F223  C9            ret

*****************************************************************
F224  05            db    05          Anzahl der Tabelleneinträge
F225  33F2          dw    F233        Rücksprungadresse falls nicht gefunden
F227  2C            db    2C          ','
F228  5CF2          dw    F25C
F22A  E5            db    E5          'SPC'
F22B  77F2          dw    F277
F22D  EA            db    EA          'TAB'
F22E  80F2          dw    F280
F230  3B            db    3B          ';'
F231  3FDD          dw    DD3F        Blanks überlesen

*********************************************************** PRINT
F233  CDFBCE        call  CEFB        Ausdruck holen
F236  F5            push  af
F237  E5            push  hl
F238  CD45FF        call  FF45        Test auf String
F23B  280C          jr    z,F249      ja
F23D  CD9DEE        call  EE9D        Zahl in ASCII-String wandeln
F240  CDDCF7        call  F7DC        Stringparameter holen
F243  3620          ld    (hl),20     '  ', Leereichen anhängen
F245  2AC2B0        ld    hl,(B0C2)
F248  34            inc   (hl)
F249  2AC2B0        ld    hl,(B0C2)
F24C  7E            ld    a,(hl)
F24D  CDB9C2        call  C2B9        Ausgabestrom selektieren
F250  D44EC3        call  nc,C34E     LF ausgeben
F253  CD28F8        call  F828        String ausgeben
F256  E1            pop   hl
F257  F1            pop   af
F258  CC4EC3        call  z,C34E      LF ausgeben
F25B  C9            ret

*********************************************************** PRINT ,
F25C  CD3FDD        call  DD3F        Blanks überlesen
F25F  3A79AE        ld    a,(AE79)    Tabulatorweite
F262  4F            ld    c,a
F263  CD90C2        call  C290
F266  3D            dec   a
F267  91            sub   c
F268  30FD          jr    nc,F267
F26A  2F            cpl   a
F26B  3C            inc   a
F26C  47            ld    b,a
F26D  81            add   a,c
F26E  CDB9C2        call  C2B9        Ausgabestrom selektieren
F271  D24EC3        jp    nc,C34E     LF ausgeben
F274  78            ld    a,b
F275  181E          jr    F295
```

```
*********************************************************** PRINT SPC
F277  CDA0F2        call  F2A0        Argument in Klammern holen
F27A  CDAFF2        call  F2AF
F27D  7B            ld    a,e
F27E  1815          jr    F295

*********************************************************** PRINT TAB
F280  CDA0F2        call  F2A0        Argument in Klammern holen
F283  1B            dec   de
F284  CDAFF2        call  F2AF
F287  CD90C2        call  C290
F28A  2F            cpl   a
F28B  3C            inc   a
F28C  1C            inc   e
F28D  83            add   a,e
F28E  3805          jr    c,F295
F290  CD4EC3        call  C34E        LF ausgeben
F293  1D            dec   e
F294  7B            ld    a,e
F295  47            ld    b,a
F296  04            inc   b
F297  05            dec   b
F298  C8            ret   z
F299  3E20          ld    a,20        ' '
F29B  CD56C3        call  C356        ausgeben
F29E  18F7          jr    F297

********************************************* Argument in Klammern holen
F2A0  CD3FDD        call  DD3F        Blanks überlesen
F2A3  CD37DD        call  DD37        Test auf nachfolgendes Zeichen
F2A6  28            db    28          '('
F2A8  CD86CE        call  CE86        Integerwert mit Vorzeichen holen
F2A9  CD37DD        call  DD37        Test auf nachfolgendes Zeichen
F2AD  29            db    29          ')'
F2AE  C9            ret

F2AF  7A            ld    a,d
F2B0  17            rla
F2B1  3003          jr    nc,F2B6
F2B3  110000        ld    de,0000
F2B6  CD9FC2        call  C29F
F2B9  D0            ret   nc
F2BA  E5            push  hl
F2BB  EB            ex    de,hl
F2BC  5F            ld    e,a         Akku als Divisor
F2BD  1600          ld    d,00        Hi-Byte Null
F2BF  CDC1BD        call  BDC1        Integerdivison ohne Vorzeichen
F2C2  E1            pop   hl          Rest in de
F2C3  C9            ret

********************************************************* PRINT USING
F2C4  CD3FDD        call  DD3F        Blanks überlesen
F2C7  CDA5CE        call  CEA5        Stringausdruck holen
F2CA  CD37DD        call  DD37        Test auf nachfolgendes Zeichen
F2CD  3B            db    3B          ';'
```

```
F2CE  E5        push  hl
F2CF  2AC2B0    ld    hl,(B0C2)
F2D2  7E        ld    a,(hl)
F2D3  B7        or    a
F2D4  2875      jr    z,F34B        'Improper argument'
F2D6  E3        ex    (sp),hl
F2D7  CDFBCE    call  CEFB          Ausdruck holen
F2DA  AF        xor   a
F2DB  327AAE    ld    (AE7A),a
F2DE  D1        pop   de
F2DF  D5        push  de
F2E0  EB        ex    de,hl
F2E1  46        ld    b,(hl)
F2E2  23        inc   hl
F2E3  7E        ld    a,(hl)
F2E4  23        inc   hl
F2E5  66        ld    h,(hl)
F2E6  6F        ld    l,a
F2E7  EB        ex    de,hl
F2E8  CD24F3    call  F324
F2EB  305E      jr    nc,F34B       'Improper argument'
F2ED  CD51DD    call  DD51          Ende des Statements ?
F2F0  381D      jr    c,F30F        ja
F2F2  FE3B      cp    3B            ';'
F2F4  2804      jr    z,F2FA
F2F6  FE2C      cp    2C            ','
F2F8  204C      jr    nz,F346       'Syntax error'
F2FA  CD3FDD    call  DD3F          Blanks überlesen
F2FD  2810      jr    z,F30F        Zeilenende ?
F2FF  D5        push  de
F300  CDFBCE    call  CEFB          Ausdruck holen
F303  D1        pop   de
F304  78        ld    a,b
F305  B7        or    a
F306  28D6      jr    z,F2DE
F308  CD24F3    call  F324
F30B  30D1      jr    nc,F2DE
F30D  18DE      jr    F2ED

F30F  F5        push  af
F310  3EFF      ld    a,FF
F312  327AAE    ld    (AE7A),a
F315  78        ld    a,b
F316  B7        or    a
F317  C424F3    call  nz,F324
F31A  F1        pop   af
F31B  DC4EC3    call  c,C34E        LF ausgeben
F31E  E3        ex    (sp),hl
F31F  CDE8FB    call  FBE8
F322  E1        pop   hl
F323  C9        ret
```

```
F324  E5        push  hl
F325  1A        ld    a,(de)
F326  FE5F      cp    5F
F328  2009      jr    nz,F333
F32A  78        ld    a,b
F32B  FE02      cp    02
F32D  380C      jr    c,F33B
F32F  13        inc   de
F330  05        dec   b
F331  1808      jr    F33B

F333  CD50F3    call  F350
F336  D4A3F3    call  nc,F3A3
F339  3809      jr    c,F344
F33B  1A        ld    a,(de)
F33C  CD56C3    call  C356        ausgeben
F33F  13        inc   de
F340  05        dec   b
F341  20E2      jr    nz,F325
F343  B7        or    a
F344  E1        pop   hl
F345  C9        ret

F346  1E02      ld    e,02        'Syntax error'
F348  C394CA    jp    CA94        Fehlermeldung ausgeben

F34B  1E05      ld    e,05        'Improper argument'
F34D  C394CA    jp    CA94        Fehlermeldung ausgeben

F350  1A        ld    a,(de)
F351  FE21      cp    21          '!'
F353  0E01      ld    c,01
F355  2821      jr    z,F378
F357  FE26      cp    26          '&'
F359  0E00      ld    c,00
F35B  281B      jr    z,F378
F35D  EE5C      xor   5C          'Backslash'
F35F  C0        ret   nz
F360  C5        push  bc
F361  D5        push  de
F362  0E02      ld    c,02
F364  13        inc   de
F365  05        dec   b
F366  280A      jr    z,F372
F368  1A        ld    a,(de)
F369  FE5C      cp    5C          'Backslash'
F36B  2809      jr    z,F376
F36D  0C        inc   c
F36E  FE20      cp    20          ' '
F370  28F2      jr    z,F364
F372  D1        pop   de
F373  C1        pop   bc
F374  B7        or    a
F375  C9        ret
```

```
F376  F1          pop   af
F377  F1          pop   af
F378  13          inc   de
F379  05          dec   b
F37A  C5          push  bc
F37B  D5          push  de
F37C  3A7AAE      ld    a,(AE7A)
F37F  B7          or    a
F380  201D        jr    nz,F39F
F382  CD3CFF      call  FF3C          Typ 'String', sonst 'Type mismatch'
F385  79          ld    a,c
F386  B7          or    a
F387  F5          push  af
F388  41          ld    b,c
F389  0E00        ld    c,00
F38B  2AC2B0      ld    hl,(B0C2)
F38E  EB          ex    de,hl
F38F  C471F9      call  nz,F971
F392  CD28F8      call  F828          String ausgeben
F395  F1          pop   af
F396  2807        jr    z,F39F
F398  2AC2B0      ld    hl,(B0C2)
F39B  96          sub   (hl)
F39C  CD95F2      call  F295
F39F  D1          pop   de
F3A0  C1          pop   bc
F3A1  37          scf
F3A2  C9          ret

F3A3  CDBAF3      call  F3BA          auf Formatierungszeichen prüfen
F3A6  D0          ret   nc
F3A7  3A7AAE      ld    a,(AE7A)
F3AA  B7          or    a
F3AB  200B        jr    nz,F3B8
F3AD  C5          push  bc
F3AE  D5          push  de
F3AF  79          ld    a,c
F3B0  CD9FEE      call  EE9F          Zahl formatieren
F3B3  CD41C3      call  C341          String bis (0) ausgeben
F3B6  D1          pop   de
F3B7  C1          pop   bc
F3B8  37          scf
F3B9  C9          ret

**************************************** auf Formatierungszeichen prüfen
F3BA  C5          push  bc
F3BB  D5          push  de
F3BC  0E80        ld    c,80
F3BE  2600        ld    h,00
F3C0  1A          ld    a,(de)
F3C1  FE2B        cp    2B            '+'
F3C3  2007        jr    nz,F3CC
F3C5  13          inc   de
F3C6  05          dec   b
F3C7  2823        jr    z,F3EC
```

```
F3C9  24          inc   h
F3CA  0E88        ld    c,88
F3CC  1A          ld    a,(de)
F3CD  FE2E        cp    2E        '.'
F3CF  281F        jr    z,F3F0
F3D1  FE23        cp    23        '#'
F3D3  2839        jr    z,F40E
F3D5  13          inc   de
F3D6  05          dec   b
F3D7  2813        jr    z,F3EC
F3D9  EB          ex    de,hl
F3DA  BE          cp    (hl)
F3DB  EB          ex    de,hl
F3DC  200E        jr    nz,F3EC
F3DE  24          inc   h
F3DF  24          inc   h
F3E0  2E04        ld    l,04
F3E2  FE24        cp    24        '$'
F3E4  2823        jr    z,F409
F3E6  2E20        ld    l,20      ' '
F3E8  FE2A        cp    2A        '*'
F3EA  2811        jr    z,F3FD
F3EC  D1          pop   de
F3ED  C1          pop   bc
F3EE  B7          or    a
F3EF  C9          ret

F3F0  13          inc   de
F3F1  05          dec   b
F3F2  28F8        jr    z,F3EC
F3F4  1A          ld    a,(de)
F3F5  FE23        cp    23        '#'
F3F7  20F3        jr    nz,F3EC
F3F9  1B          dec   de
F3FA  04          inc   b
F3FB  1811        jr    F40E

F3FD  13          inc   de
F3FE  05          dec   b
F3FF  280A        jr    z,F40B
F401  1A          ld    a,(de)
F402  FE24        cp    24        '$'
F404  2005        jr    nz,F40B
F406  24          inc   h
F407  2E24        ld    l,24
F409  13          inc   de
F40A  05          dec   b
F40B  79          ld    a,c
F40C  B5          or    l
F40D  4F          ld    c,a
F40E  F1          pop   af
F40F  F1          pop   af
F410  CD1BF4      call  F41B
F413  7C          ld    a,h
F414  85          add   a,l
```

```
F415  FE15      cp    15
F417  D24BF3    jp    nc,F34B        'Improper argument'
F41A  C9        ret

F41B  2E00      ld    l,00
F41D  04        inc   b
F41E  05        dec   b
F41F  C8        ret   z
F420  1A        ld    a,(de)
F421  FE2E      cp    2E             '.'
F423  2814      jr    z,F439
F425  FE2C      cp    2C             ','
F427  280A      jr    z,F433
F429  FE23      cp    23             '#'
F42B  2015      jr    nz,F442
F42D  24        inc   h
F42E  13        inc   de
F42F  05        dec   b
F430  20EE      jr    nz,F420
F432  C9        ret

F433  79        ld    a,c
F434  F602      or    02
F436  4F        ld    c,a
F437  18F4      jr    F42D

F439  2C        inc   l
F43A  13        inc   de
F43B  05        dec   b
F43C  C8        ret   z
F43D  1A        ld    a,(de)
F43E  FE23      cp    23             '#'
F440  28F7      jr    z,F439
F442  EB        ex    de,hl
F443  E5        push  hl
F444  FE5E      cp    5E             '↑'
F446  2018      jr    nz,F460
F448  23        inc   hl
F449  BE        cp    (hl)
F44A  2014      jr    nz,F460
F44C  23        inc   hl
F44D  BE        cp    (hl)
F44E  2010      jr    nz,F460
F450  23        inc   hl
F451  BE        cp    (hl)
F452  200C      jr    nz,F460
F454  23        inc   hl
F455  78        ld    a,b
F456  D604      sub   04
F458  3806      jr    c,F460
F45A  47        ld    b,a
F45B  E3        ex    (sp),hl
F45C  79        ld    a,c
F45D  F640      or    40
F45F  4F        ld    c,a
```

```
F460  E1          pop   hl
F461  EB          ex    de,hl
F462  78          ld    a,b
F463  B7          or    a
F464  C8          ret   z
F465  79          ld    a,c
F466  E608        and   08
F468  C0          ret   nz
F469  1A          ld    a,(de)
F46A  FE2D        cp    2D          '-'
F46C  3E10        ld    a,10
F46E  2806        jr    z,F476
F470  1A          ld    a,(de)
F471  FE2B        cp    2B          '+'
F473  C0          ret   nz
F474  3E18        ld    a,18
F476  B1          or    c
F477  4F          ld    c,a
F478  13          inc   de
F479  05          dec   b
F47A  C9          ret

*********************************************** BASIC-Befehl WRITE
F47B  CDC6C1      call  C1C6        Kanalnummer vorhanden ?
F47E  F5          push  af
F47F  CD51DD      call  DD51        Ende des Statements ?
F482  3839        jr    c,F4BD      ja
F484  CDFBCE      call  CEFB        Ausdruck holen
F487  F5          push  af
F488  E5          push  hl
F489  CD45FF      call  FF45        Test auf String
F48C  280B        jr    z,F499      ja, mit Hochkommas ausgeben
F48E  CD8FEE      call  EE8F
F491  CDDCF7      call  F7DC
F494  CD28F8      call  F828        String ausgeben
F497  180D        jr    F4A6

F499  3E22        ld    a,22        '"'
F49B  CD56C3      call  C356        ausgeben
F49E  CD28F8      call  F828        String ausgeben
F4A1  3E22        ld    a,22        '"'
F4A3  CD56C3      call  C356        ausgeben
F4A6  E1          pop   hl
F4A7  F1          pop   af
F4A8  2813        jr    z,F4BD
F4AA  FE3B        cp    3B          ';'
F4AC  2805        jr    z,F4B3
F4AE  FE2C        cp    2C          ','
F4B0  C246F3      jp    nz,F346     'Syntax error'
F4B3  CD3FDD      call  DD3F        Blanks überlesen
F4B6  3E2C        ld    a,2C        ','
F4B8  CD56C3      call  C356        ausgeben
F4BB  18C7        jr    F484
```

```
F4BD  CD4EC3     call  C34E        LF ausgeben
F4C0  F1         pop   af
F4C1  C3A2C1     jp    C1A2

*************************************************** Speicher konfigurieren
F4C4  0100AC     ld    bc,AC00     Speicherplatz von de bis hl
F4C7  CDBEFF     call  FFBE        Vergleich hl <> bc
F4CA  D0         ret   nc          höchste Adresse < AC00 ?
F4CB  227BAE     ld    (AE7B),hl   HIMEM
F4CE  228FB0     ld    (B08F),hl   Ende der Strings
F4D1  227DAE     ld    (AE7D),hl   Ende des freien RAMs
F4D4  EB         ex    de,hl
F4D5  227FAE     ld    (AE7F),hl   Beginn des freien RAMs
F4D8  012F01     ld    bc,012F     plus 303
F4DB  09         add   hl,bc
F4DC  D8         ret   c
F4DD  2281AE     ld    (AE81),hl   ergibt Programmstart
F4E0  EB         ex    de,hl
F4E1  23         inc   hl
F4E2  B7         or    a
F4E3  ED52       sbc   hl,de
F4E5  D8         ret   c
F4E6  7C         ld    a,h
F4E7  FE04       cp    04
F4E9  D8         ret   c
F4EA  AF         xor   a
F4EB  3291B0     ld    (B091),a
F4EE  C9         ret

*************************************************** BASIC-Befehl MEMORY
F4EF  CD3EFC     call  FC3E        Garbage Collection
F4F2  CD91CE     call  CE91        16-Bit-Wert holen
F4F5  E5         push  hl
F4F6  CD50F7     call  F750
F4F9  CD75F6     call  F675
F4FC  227BAE     ld    (AE7B),hl   HIMEM setzen
F4FF  E1         pop   hl
F500  C9         ret

********************************************* Platz für zu ladendes Programm
F501  D5         push  de
F502  2A7FAE     ld    hl,(AE7F)   Beginn des freien RAMs
F505  EB         ex    de,hl
F506  2A7BAE     ld    hl,(AE7B)   HIMEM
F509  CDCFFF     call  FFCF        hl := hl - de
F50C  E3         ex    (sp),hl
F50D  CDCFFF     call  FFCF        hl := hl - de
F510  D1         pop   de
F511  13         inc   de
F512  CDB8FF     call  FFB8        Vergleich hl <> de
F515  3803       jr    c,F51A      'Memory full'
F517  2B         dec   hl
F518  09         add   hl,bc
F519  D0         ret   nc
F51A  C33EF7     jp    F73E        'Memory full'
```

```
************************************ L41nge des Stringbereichs berechnen
F51D  D5        push  de
F51E  E5        push  hl
F51F  2A8DB0    ld    hl,(B08D)     Beginn der Strings
F522  EB        ex    de,hl
F523  2A8FB0    ld    hl,(B08F)     Ende der Strings
F526  CDDAFF    call  FFDA          bc := hl - de
F529  E1        pop   hl
F52A  D1        pop   de
F52B  C9        ret

************************************ Prg- und Variablenzeiger um bc erhöhen
F52C  2A83AE    ld    hl,(AE83)     Programmende
F52F  09        add   hl,bc
F530  2283AE    ld    (AE83),hl     Programmende
F533  2A85AE    ld    hl,(AE85)     Variablenstart
F536  09        add   hl,bc
F537  2285AE    ld    (AE85),hl     Variablenstart
F53A  2A87AE    ld    hl,(AE87)     Arraystart
F53D  09        add   hl,bc
F53E  2287AE    ld    (AE87),hl     Arraystart
F541  2A89AE    ld    hl,(AE89)     Arrayende
F544  09        add   hl,bc
F545  2289AE    ld    (AE89),hl     Arrayende
F548  C9        ret

************************************************************
F549  2A85AE    ld    hl,(AE85)     Variablenstart
F54C  EB        ex    de,hl
F54D  2A87AE    ld    hl,(AE87)     Arraystart
F550  CDCFFF    call  FFCF          hl := hl - de
F553  E5        push  hl
F554  2A89AE    ld    hl,(AE89)     Arrayende
F557  CDDAFF    call  FFDA          bc := hl - de
F55A  C5        push  bc
F55B  2A8DB0    ld    hl,(B08D)     Beginn der Strings
F55E  EB        ex    de,hl
F55F  2A89AE    ld    hl,(AE89)     Arrayende
F562  2B        dec   hl
F563  78        ld    a,b
F564  B1        or    c
F565  C4F5FF    call  nz,FFF5       lddr
F568  EB        ex    de,hl
F569  228DB0    ld    (B08D),hl     Beginn der Strings
F56C  C1        pop   bc
F56D  D1        pop   de
F56E  C3B1D5    jp    D5B1          Variablenzeiger zurücksetzen

************************************************************
F571  2A83AE    ld    hl,(AE83)     Programmende
F574  2285AE    ld    (AE85),hl     gleich Variablenstart
F577  EB        ex    de,hl
F578  19        add   hl,de         plus Variablenl41nge
F579  2287AE    ld    (AE87),hl     gleich Arraystart
F57C  2A8DB0    ld    hl,(B08D)     Beginn der Strings
```

```
F57F  23        inc   hl
F580  78        ld    a,b          Stringbreich
F581  B1        or    c
F582  C4F2FF    call  nz,FFF2      vorhanden, dann ldir
F585  2B        dec   hl
F586  228DB0    ld    (B08D),hl    Beginn der Strings
F589  EB        ex    de,hl
F58A  2289AE    ld    (AE89),hl    Arrayende
F58D  C9        ret

*************************************** BASIC-Stack initialisieren
F58E  F5        push  af
F58F  E5        push  hl
F590  218BAE    ld    hl,AE8B
F593  228BB0    ld    (B08B),hl    BASIC-Stackpointer
F596  3E01      ld    a,01
F598  CDB0F5    call  F5B0         Platz im BASIC-Stack reservieren
F59B  3600      ld    (hl),00
F59D  E1        pop   hl
F59E  F1        pop   af
F59F  C9        ret

************************************* Platz im BASIC-Stack freigeben
F5A0  2A8BB0    ld    hl,(B08B)    BASIC-Stackpointer
F5A3  2F        cpl   a
F5A4  3C        inc   a            Akkuinhalt abziehen
F5A5  C8        ret   z
F5A6  85        add   a,l
F5A7  6F        ld    l,a
F5A8  3EFF      ld    a,FF
F5AA  8C        adc   a,h
F5AB  67        ld    h,a
F5AC  228BB0    ld    (B08B),hl    BASIC-Stackpointer
F5AF  C9        ret

************************************ Platz im BASIC-Stack reservieren
F5B0  2A8BB0    ld    hl,(B08B)    BASIC-Stackpointer
F5B3  E5        push  hl
F5B4  85        add   a,l
F5B5  6F        ld    l,a          Akkuinhalt addieren
F5B6  8C        adc   a,h
F5B7  95        sub   l
F5B8  67        ld    h,a
F5B9  228BB0    ld    (B08B),hl    BASIC-Stackpointer
F5BC  3E78      ld    a,78
F5BE  85        add   a,l          ergibt plus &4F78 Überlauf ?
F5BF  3E4F      ld    a,4F
F5C1  8C        adc   a,h          dann ist Stackpointer > &B088
F5C2  E1        pop   hl
F5C3  D0        ret   nc
F5C4  CD8EF5    call  F58E         BASIC-Stack initialisieren
F5C7  C33EF7    jp    F73E         'Memory full'
```

```
*********************************************************************
F5CA  2A8FB0     ld    hl,(B08F)      Ende der Strings
F5CD  228DB0     ld    (B08D),hl      Beginn der Strings
F5D0  C9         ret

******************************************************* Platz für String reservieren
F5D1  2F         cpl   a              Akku enthält Stringlänge
F5D2  4F         ld    c,a
F5D3  06FF       ld    b,FF           minus Länge nach bc
F5D5  03         inc   bc
F5D6  CDE6F5     call  F5E6           Stringbereich nach unten verlängern
F5D9  D0         ret   nc             Platz vorhanden ?
F5DA  CD3EFC     call  FC3E           nein, Garbage Collection auslösen
F5DD  CDE6F5     call  F5E6           jetzt Platz vorhanden ?
F5E0  D0         ret   nc             ja
F5E1  1E0E       ld    e,0E           'String space full'
F5E3  C394CA     jp    CA94           Fehlermeldung ausgeben

******************************************************** Platz im Stringbereich ?
F5E6  2A89AE     ld    hl,(AE89)      Arrayende
F5E9  EB         ex    de,hl
F5EA  2A8DB0     ld    hl,(B08D)      Beginn der Strings
F5ED  09         add   hl,bc          minus Länge des neuen Strings
F5EE  CDB8FF     call  FFB8           Vergleich hl <> de
F5F1  D8         ret   c
F5F2  228DB0     ld    (B08D),hl      Beginn der Strings
F5F5  23         inc   hl
F5F6  EB         ex    de,hl          nach de
F5F7  C9         ret

********************************************** Platz im Variablenbereich reservieren
F5F8  2A89AE     ld    hl,(AE89)      Arrayende
F5FB  C5         push  bc             enthält Anzahl Bytes
F5FC  D5         push  de
F5FD  D5         push  de
F5FE  E5         push  hl
F5FF  CD18F6     call  F618           Platz vorhanden ?
F602  DA3EF7     jp    c,F73E         nein, 'Memory full'
F605  E1         pop   hl
F606  C1         pop   bc
F607  D5         push  de
F608  7D         ld    a,l
F609  91         sub   c
F60A  4F         ld    c,a
F60B  7C         ld    a,h
F60C  98         sbc   a,b
F60D  47         ld    b,a
F60E  2B         dec   hl
F60F  1B         dec   de
F610  B1         or    c
F611  C4F5FF     call  nz,FFF5        lddr
F614  E1         pop   hl
F615  D1         pop   de
F616  C1         pop   bc
F617  C9         ret
```

```
*************************************** Platz im Variablenbereich vorhanden ?
F618  09        add   hl,bc        Ende Array plus neuer Platz
F619  D8        ret   c            Überlauf ?
F61A  EB        ex    de,hl
F61B  CD22F6    call  F622         neues Variablenende mit Stringbeginn vergl
F61E  D0        ret   nc           Platz vorhanden ?
F61F  CD3EFC    call  FC3E         nein, Garbage Collection auslösen
F622  2A8DB0    ld    hl,(B08D)    Beginn der Strings
F625  C3B8FF    jp    FFB8         Vergleich hl <> de

**************************************** freien Speicherplatz berechnen
F628  2A89AE    ld    hl,(AE89)    Ende der Variablen
F62B  EB        ex    de,hl
F62C  2A8DB0    ld    hl,(B08D)    Beginn der Strings
F62F  C3CFFF    jp    FFFC         hl := hl - de

********************************************** Eingabepuffer anlegen
F632  110100    ld    de,0001
F635  1803      jr    F63A

********************************************** Ausgabepuffer anlegen
F637  110208    ld    de,0802
F63A  C5        push  bc
F63B  E5        push  hl
F63C  2191B0    ld    hl,B091
F63F  7E        ld    a,(hl)
F640  B7        or    a
F641  201D      jr    nz,F660
F643  D5        push  de
F644  E5        push  hl
F645  210010    ld    hl,1000
F648  010000    ld    bc,0000
F64B  CD43F7    call  F743
F64E  2292B0    ld    (B092),hl    neues Ende des freien RAM
F651  EB        ex    de,hl
F652  2A7DAE    ld    hl,(AE7D)    Ende des freien RAMs
F655  2294B0    ld    (B094),hl    Speicher für freies RAM
F658  EB        ex    de,hl
F659  227DAE    ld    (AE7D),hl    Ende des freien RAMs
F65C  E1        pop   hl
F65D  D1        pop   de
F65E  3E04      ld    a,04
F660  B3        or    e
F661  77        ld    (hl),a
F662  2A92B0    ld    hl,(B092)    neues Ende des freien RAM
F665  23        inc   hl
F666  1E00      ld    e,00
F668  19        add   hl,de
F669  EB        ex    de,hl
F66A  E1        pop   hl
F66B  C1        pop   bc
F66C  C9        ret
```

```
************************************************ Eingabepuffer schließen
F66D  3EFE      ld    a,FE
F66F  1806      jr    F677

************************************************ Ausgabepuufer schließen
F671  3EFD      ld    a,FD
F673  1802      jr    F677

*********************************************************************
F675  3EFF      ld    a,FF
F677  C5        push  bc
F678  D5        push  de
F679  E5        push  hl
F67A  2191B0    ld    hl,B091
F67D  A6        and   (hl)
F67E  77        ld    (hl),a
F67F  FE04      cp    04
F681  2016      jr    nz,F699
F683  2A92B0    ld    hl,(B092)     neues Ende des freien RAM
F686  EB        ex    de,hl
F687  210010    ld    hl,1000
F68A  CD2EF7    call  F72E
F68D  200A      jr    nz,F699
F68F  AF        xor   a
F690  3291B0    ld    (B091),a
F693  2A94B0    ld    hl,(B094)     Speicher für freies RAM
F696  227DAE    ld    (AE7D),hl     Ende des freien RAMs
F699  E1        pop   hl
F69A  D1        pop   de
F69B  C1        pop   bc
F69C  C9        ret

************************************************ BASIC-Befehl SYMBOL
F69D  FE80      cp    80            'AFTER'
F69F  282C      jr    z,F6CD
F6A1  CD67CE    call  CE67          8-Bit-Wert holen
F6A4  4F        ld    c,a
F6A5  CD37DD    call  DD37          Test auf nachfolgendes Zeichen
F6A8  2C        db    2C            ','
F6A9  0608      ld    b,08          8 Werte
F6AB  CD67CE    call  CE67          8-Bit-Wert holen
F6AE  F5        push  af            auf Stack
F6AF  05        dec   b
F6B0  2808      jr    z,F6BA        schon 8 Werte ?
F6B2  CD55DD    call  DD55          folgt Komma ?
F6B5  38F4      jr    c,F6AB        ja, nächsten Wert holen
F6B7  AF        xor   a
F6B8  18F4      jr    F6AE

F6BA  EB        ex    de,hl         hl retten
F6BB  79        ld    a,c           Zeichen nach a
F6BC  CDA5BB    call  BBA5          TXT GET MATRIX
F6BF  3068      jr    nc,F729       Matrix nicht im RAM, 'Improper Argument'
F6C1  010800    ld    bc,0008       8
F6C4  09        add   hl,bc         plus Matrixadresse
```

```
F6C5  F1          pop   af          Byte vom Stack holen
F6C6  2B          dec   hl
F6C7  77          ld    (hl),a      in Matrixtabelle schreiben
F6C8  0D          dec   c           nächstes Byte
F6C9  20FA        jr    nz,F6C5
F6CB  EB          ex    de,hl       hl zurück
F6CC  C9          ret

*********************************************** SYMBOL AFTER
F6CD  CD3FDD      call  DD3F        Blanks überlesen
F6D0  CD86CE      call  CE86        Integerwert mit Vorzeichen holen
F6D3  E5          push  hl
F6D4  210001      ld    hl,0100     256
F6D7  CDB8FF      call  FFB8        Vergleich hl <> de
F6DA  384D        jr    c,F729      größer gleich 256, 'Improper argument'
F6DC  D5          push  de
F6DD  CDAEBB      call  BBAE        TXT GET M TABLE
F6E0  EB          ex    de,hl       Matrixadresse nach de
F6E1  301D        jr    nc,F700     Matrix noch nicht definiert ?
F6E3  2F          cpl   a
F6E4  6F          ld    l,a
F6E5  2600        ld    h,00
F6E7  23          inc   hl
F6E8  29          add   hl,hl
F6E9  29          add   hl,hl
F6EA  29          add   hl,hl
F6EB  1B          dec   de
F6EC  CD2EF7      call  F72E
F6EF  2038        jr    nz,F729     'Improper argument'
F6F1  2A96B0      ld    hl,(B096)
F6F4  227DAE      ld    (AE7D),hl   Ende des freien RAMs
F6F7  CD75F6      call  F675
F6FA  110001      ld    de,0100
F6FD  CDABBB      call  BBAB        TXT SET M TABLE
F700  D1          pop   de
F701  CD06F7      call  F706
F704  E1          pop   hl
F705  C9          ret

F706  AF          xor   a
F707  93          sub   e
F708  6F          ld    l,a
F709  3E01        ld    a,01
F70B  9A          sbc   a,d
F70C  67          ld    h,a
F70D  B5          or    l
F70E  C8          ret   z
F70F  D5          push  de          erstes Zeichen merken
F710  29          add   hl,hl
F711  29          add   hl,hl
F712  29          add   hl,hl
F713  010040      ld    bc,4000
F716  CD43F7      call  F743
F719  EB          ex    de,hl
F71A  2A7DAE      ld    hl,(AE7D)   Ende des freien RAMs
```

```
F71D  2296B0     ld    (B096),hl
F720  EB         ex    de,hl
F721  227DAE     ld    (AE7D),hl      Ende des freien RAMs
F724  D1         pop   de             erstes Zeichen
F725  23         inc   hl             Startadresse der Tabelle
F726  C3ABBB     jp    BBAB           TXT SET M TABLE

F729  1E05       ld    e,05           'Improper argument'
F72B  C394CA     jp    CA94           Fehlermeldung ausgeben

F72E  E5         push  hl
F72F  2A7BAE     ld    hl,(AE7B)      HIMEM
F732  CDB8FF     call  FFB8           Vergleich hl <> de
F735  E1         pop   hl
F736  C0         ret   nz
F737  19         add   hl,de
F738  227DAE     ld    (AE7D),hl      Ende des freien RAMs
F73B  EB         ex    de,hl
F73C  1812       jr    F750

F73E  1E07       ld    e,07           'Memory full'
F740  C394CA     jp    CA94           Fehlermeldung ausgeben

F743  EB         ex    de,hl
F744  2A7BAE     ld    hl,(AE7B)      HIMEM
F747  CDCFFF     call  FFCF           hl := hl - de
F74A  CDBEFF     call  FFBE           Vergleich hl <> bc
F74D  38EF       jr    c,F73E         'Memory full'
F74F  EB         ex    de,hl
F750  CD3EFC     call  FC3E           Garbage Collection
F753  D5         push  de
F754  2A7DAE     ld    hl,(AE7D)      Ende des freien RAMs
F757  CDB8FF     call  FFB8           Vergleich hl <> de
F75A  38E2       jr    c,F73E         'Memory full'
F75C  CD1DF5     call  F51D           Länge des Stringbereichs berechnen
F75F  2A89AE     ld    hl,(AE89)      Arrayende
F762  09         add   hl,bc          plus Länge des Stringbereichs
F763  38D9       jr    c,F73E         'Memory full'
F765  2B         dec   hl
F766  CDB8FF     call  FFB8           Vergleich hl <> de
F769  30D3       jr    nc,F73E        'Memory full'
F76B  2A7BAE     ld    hl,(AE7B)      HIMEM
F76E  EB         ex    de,hl
F76F  CDCFFF     call  FFCF           hl := hl - de
F772  2298B0     ld    (B098),hl
F775  11BBF7     ld    de,F7BB
F778  CD74DA     call  DA74
F77B  ED4B98B0   ld    bc,(B098)
F77F  78         ld    a,b
F780  07         rlca
F781  3816       jr    c,F799
F783  B1         or    c
F784  282F       jr    z,F7B5
F786  2A8FB0     ld    hl,(B08F)      Ende der Strings
F789  54         ld    d,h
```

```
F78A  5D          ld     e,l
F78B  09          add    hl,bc
F78C  E5          push   hl
F78D  CD1DF5      call   F51D          Länge des Stringbereichs berechnen
F790  EB          ex     de,hl
F791  78          ld     a,b
F792  B1          or     c
F793  C4F5FF      call   nz,FFF5       lddr
F796  E1          pop    hl
F797  1815        jr     F7AE

F799  2A8DB0      ld     hl,(B08D)     Beginn der Strings
F79C  54          ld     d,h
F79D  5D          ld     e,l
F79E  09          add    hl,bc
F79F  E5          push   hl
F7A0  CD1DF5      call   F51D          Länge des Stringbereichs berechnen
F7A3  EB          ex     de,hl
F7A4  23          inc    hl
F7A5  13          inc    de
F7A6  78          ld     a,b
F7A7  B1          or     c
F7A8  C4F2FF      call   nz,FFF2       ldir
F7AB  EB          ex     de,hl
F7AC  2B          dec    hl
F7AD  D1          pop    de
F7AE  228FB0      ld     (B08F),hl     Ende der Strings
F7B1  EB          ex     de,hl
F7B2  228DB0      ld     (B08D),hl     Beginn der Strings
F7B5  E1          pop    hl
F7B6  227BAE      ld     (AE7B),hl     HIMEM
F7B9  AF          xor    a
F7BA  C9          ret

F7BB  2A83AE      ld     hl,(AE83)     Programmende
F7BE  CDBEFF      call   FFBE          Vergleich hl <> bc
F7C1  D0          ret    nc
F7C2  2A98B0      ld     hl,(B098)
F7C5  09          add    hl,bc
F7C6  EB          ex     de,hl
F7C7  72          ld     (hl),d
F7C8  2B          dec    hl
F7C9  73          ld     (hl),e
F7CA  C9          ret

****************************************************** String lesen
F7CB  23          inc    hl
F7CC  CDF9F7      call   F7F9
F7CF  7E          ld     a,(hl)
F7D0  FE22        cp     22            '"', Stringende ?
F7D2  CA3FDD      jp     z,DD3F        ja, folgende Blanks überlesen
F7D5  B7          or     a
F7D6  2837        jr     z,F80F
F7D8  04          inc    b
F7D9  23          inc    hl
```

```
F7DA  18F3       jr     F7CF

F7DC  CDF9F7     call   F7F9
F7DF  7E         ld     a,(hl)
F7E0  B7         or     a
F7E1  C8         ret    z
F7E2  23         inc    hl
F7E3  04         inc    b
F7E4  18F9       jr     F7DF

F7E6  CDF9F7     call   F7F9
F7E9  4F         ld     c,a
F7EA  7E         ld     a,(hl)
F7EB  B7         or     a
F7EC  2821       jr     z,F80F
F7EE  B9         cp     c
F7EF  281E       jr     z,F80F
F7F1  FE2C       cp     2C          ','
F7F3  281A       jr     z,F80F
F7F5  23         inc    hl
F7F6  04         inc    b
F7F7  18F1       jr     F7EA

*********************************************************************
F7F9  D1         pop    de
F7FA  E5         push   hl
F7FB  0600       ld     b,00
F7FD  CDFBFF     call   FFFB        jp (de)
F800  D1         pop    de
F801  E5         push   hl
F802  21BAB0     ld     hl,B0BA     Zeiger auf Descriptorstack
F805  70         ld     (hl),b      Länge
F806  23         inc    hl
F807  73         ld     (hl),e
F808  23         inc    hl          Adresse
F809  72         ld     (hl),d
F80A  CDBAFB     call   FBBA
F80D  E1         pop    hl
F80E  C9         ret

F80F  E5         push   hl
F810  04         inc    b
F811  05         dec    b
F812  2812       jr     z,F826
F814  2B         dec    hl
F815  7E         ld     a,(hl)
F816  FE20       cp     20          ' '
F818  28F7       jr     z,F811
F81A  FE09       cp     09          TAB
F81C  28F3       jr     z,F811
F81E  FE0D       cp     0D          CR
F820  28EF       jr     z,F811
F822  FE0A       cp     0A          LF
F824  28EB       jr     z,F811
F826  E1         pop    hl
```

```
F827  C9         ret

************************************************* String ausgeben
F828  CDDAFB     call   FBDA          Stringparameter holen
F82B  C8         ret    z             Leerstring ?
F82C  1A         ld     a,(de)        Zeichen holen
F82D  13         inc    de            Zeiger erhöhen
F82E  CD6EC3     call   C36E          Zeichen ausgeben
F831  10F9       djnz   F82C          nächstes Zeichen
F833  C9         ret

******************************************** BASIC-Funktion LOWER$
F834  0139F8     ld     bc,F839       Groß- in Kleinbuchstaben wandeln
F837  180C       jr     F845

*********************************** Umwandlung Groß- in Kleinbuchstaben
F839  FE41       cp     41            'A'
F83B  D8         ret    c
F83C  FE5B       cp     5B            'Z'+1
F83E  D0         ret    nc
F83F  C620       add    a,20          'a'-'A'
F841  C9         ret

******************************************** BASIC-Funktion UPPER$
F842  018AFF     ld     bc,FF8A       Klein- in Großbuchstaben wandeln
F845  C5         push   bc
F846  2AC2B0     ld     hl,(B0C2)
F849  7E         ld     a,(hl)        Stringlänge
F84A  CD19FC     call   FC19          Platz reservieren, Stringdescriptor ablegen
F84D  D5         push   de
F84E  CDDAFB     call   FBDA          Stringparameter holen
F851  E1         pop    hl
F852  C1         pop    bc
F853  3C         inc    a
F854  3D         dec    a
F855  CABAFB     jp     z,FBBA
F858  F5         push   af
F859  1A         ld     a,(de)
F85A  13         inc    de
F85B  CDF9FF     call   FFF9          jp (bc), Umwandlung ausführen
F85E  77         ld     (hl),a
F85F  23         inc    hl
F860  F1         pop    af
F861  18F1       jr     F854

************************************************** Stringaddition
F863  E5         push   hl
F864  7E         ld     a,(hl)        Stringlänge
F865  2AC2B0     ld     hl,(B0C2)
F868  86         add    a,(hl)        plus Länge des zweiten Strings
F869  1E0F       ld     e,0F          'String too long'
F86B  DA94CA     jp     c,CA94        Fehlermeldung ausgeben
```

```
F86E  CD19FC      call  FC19       Platz reservieren, Stringdescriptor ablegen
F871  E1          pop   hl
F872  D5          push  de
F873  E5          push  hl
F874  CDDAFB      call  FBDA       Stringparameter holen
F877  48          ld    c,b
F878  EB          ex    de,hl
F879  E3          ex    (sp),hl
F87A  CDE8FB      call  FBE8
F87D  E1          pop   hl
F87E  E3          ex    (sp),hl
F87F  78          ld    a,b
F880  CD8BF8      call  F88B
F883  D1          pop   de
F884  79          ld    a,c
F885  CD8BF8      call  F88B
F888  C3BAFB      jp    FBBA

F88B  C5          push  bc
F88C  EB          ex    de,hl
F88D  4F          ld    c,a
F88E  0600        ld    b,00
F890  B7          or    a
F891  C4F2FF      call  nz,FFF2    ldir
F894  EB          ex    de,hl
F895  C1          pop   bc
F896  C9          ret

*********************************************** Stringvergleich
F897  E5          push  hl
F898  CDDAFB      call  FBDA       Stringparameter holen
F89B  48          ld    c,b
F89C  E1          pop   hl
F89D  D5          push  de
F89E  CDE8FB      call  FBE8
F8A1  E1          pop   hl
F8A2  78          ld    a,b
F8A3  B1          or    c
F8A4  C8          ret   z
F8A5  79          ld    a,c
F8A6  B7          or    a
F8A7  280C        jr    z,F8B5
F8A9  78          ld    a,b
F8AA  B7          or    a
F8AB  2809        jr    z,F8B6
F8AD  05          dec   b
F8AE  0D          dec   c
F8AF  1A          ld    a,(de)     Zeichen aus erstem String
F8B0  13          inc   de
F8B1  BE          cp    (hl)       mit zweitem String vergleichen
F8B2  23          inc   hl
F8B3  28ED        jr    z,F8A2     gleich, dann weiter vergleichen
F8B5  3F          ccf
F8B6  9F          sbc   a,a        Flags für Ergebnis setzen
F8B7  C0          ret   nz
```

```
F8B8  3C            inc    a
F8B9  C9            ret

********************************************* BASIC-Funktion BIN$
F8BA  CDCEF8        call   F8CE         Argumente holen
F8BD  D5            push   de
F8BE  CD14F1        call   F114         in Binärstring wandeln
F8C1  EB            ex     de,hl
F8C2  185E          jr     F922         String übernehmen

********************************************* BASIC-Funktion HEX$
F8C4  CDCEF8        call   F8CE         Argumente holen
F8C7  D5            push   de
F8C8  CD19F1        call   F119         in Hexstring wandeln
F8CB  EB            ex     de,hl
F8CC  1854          jr     F92          String übernehmen

************************************** Argument für BIN$ und HEX$ holen
F8CE  CDFBCE        call   CEFB         Ausdruck holen
F8D1  CD53FF        call   FF53         und auf BASIC-Stack ablegen
F8D4  CD55DD        call   DD55         folgt Komma ?
F8D7  9F            sbc    a,a          0 als Default
F8D8  DC67CE        call   c,CE67       ja, 8-Bit-Wert holen
F8DB  FE11          cp     11           größer gleich 17 ?
F8DD  D29CFA        jp     nc,FA9C      'Improper argument'
F8E0  47            ld     b,a
F8E1  CD37DD        call   DD37         Test auf nachfolgendes Zeichen
F8E4  29            db     29           ')'
F8E5  EB            ex     hl,de
F8E6  79            ld     a,c
F8E7  C3A0F5        jp     F5A0         Platz im BASIC-Stack freigeben

******************************************** BASIC-Funktion DEC$
F8EA  CD37DD        call   DD37         Test auf nachfolgendes Zeichen
F8ED  28            db     28           '(', bereits mit Funktionsaufruf geschehen!
F8EE  CDFBCE        call   CEFB         Ausdruck holen
F8F1  CD37DD        call   DD37         Test auf nachfolgendes Zeichen
F8F4  2C            db     2C           ','
F8F5  CD53FF        call   FF53         und auf BASIC-Stack ablegen
F8F8  CD9FCE        call   CE9F         Stringausdruck und -Parameter holen
F8FB  CD37DD        call   DD37         Test auf nachfolgendes Zeichen
F8FE  29            db     29           ')'
F8FF  E5            push   hl
F900  79            ld     a,c          Länge
F901  CDA0F5        call   F5A0         Platz im BASIC-Stack freigeben
F904  D5            push   de
F905  79            ld     a,c          Länge
F906  CD4BFF        call   FF4B         Variable übernehmen
F909  D1            pop    de
F90A  78            ld     a,b
F90B  B7            or     a
F90C  C4BAF3        call   nz,F3BA      auf Formatierungszeichen prüfen
F90F  300A          jr     nc,F91B      'Improper argument'
F911  78            ld     a,b
F912  B7            or     a
```

```
F913  2006      jr    nz,F91B    'Improper argument'
F915  79        ld    a,c
F916  CD9FEE    call  EE9F       Zahl formatieren
F919  1807      jr    F922       String übernehmen

F91B  C39CFA    jp    FA9C       'Improper argument'

*************************************** BASIC-Funktion STR$
F91E  E5        push  hl
F91F  CD9DEE    call  EE9D       Zahl in String wandeln
F922  E5        push  hl
F923  01FFFF    ld    bc,FFFF    Zähler für Stringlänge auf -1
F926  03        inc   bc         Zähler erhöhen
F927  7E        ld    a,(hl)     Zeichen holen
F928  23        inc   hl
F929  B7        or    a          Nullbyte ?
F92A  20FA      jr    nz,F926    nein, nächstes Zeichen
F92C  79        ld    a,c        Stringlänge nach a
F92D  CD19FC    call  FC19       Platz reservieren, Stringdescriptor ablegen
F930  E1        pop   hl
F931  B7        or    a
F932  D5        push  de
F933  C4F2FF    call  nz,FFF2    ldir
F936  D1        pop   de
F937  CDBAFB    call  FBBA
F93A  E1        pop   hl
F93B  C9        ret

*************************************** BASIC-Funktion LEFT$
F93C  CDE9F9    call  F9E9       String und 8-Bit-Zahl holen
F93F  0E00      ld    c,00       ab Position 0
F941  182A      jr    F96D

*************************************** BASIC-Funktion RIGHT$
F943  CDE9F9    call  F9E9       String und 8-Bit-Zahl holen
F946  1A        ld    a,(de)     Stringlänge
F947  90        sub   b          minus Parameter
F948  4F        ld    c,a        ergibt Startposition
F949  1822      jr    F96D

*************************************** BASIC-Funktion MID$
F94B  CD37DD    call  DD37       Test auf nachfolgendes Zeichen
F94E  28        db    28         '('
F94F  CDE9F9    call  F9E9       String und 8-Bit-Zahl holen
F952  78        ld    a,b
F953  B7        or    a
F954  CA9CFA    jp    z,FA9C     'Improper argument'
F957  05        dec   b
F958  48        ld    c,b
F959  D5        push  de
F95A  C5        push  bc
F95B  CDFBF9    call  F9FB       3. Argument holen (Default = 255)
F95E  C1        pop   bc
F95F  E3        ex    (sp),hl
F960  7E        ld    a,(hl)
```

```
F961  91          sub   c
F962  0600        ld    b,00
F964  3805        jr    c,F96B
F966  BB          cp    e
F967  47          ld    b,a
F968  3801        jr    c,F96B
F96A  43          ld    b,e
F96B  EB          ex    de,hl
F96C  E1          pop   hl
F96D  CD37DD      call  DD37        Test auf nachfolgendes Zeichen
F970  29          db    29          ')'
F971  E5          push  hl
F972  EB          ex    de,hl
F973  7E          ld    a,(hl)
F974  B8          cp    b
F975  78          ld    a,b
F976  3003        jr    nc,F97B
F978  7E          ld    a,(hl)
F979  0E00        ld    c,00
F97B  F5          push  af
F97C  CD19FC      call  FC19        Platz reservieren, Stringdescriptor ablegen
F97F  D5          push  de
F980  CDE8FB      call  FBE8
F983  EB          ex    de,hl
F984  D1          pop   de
F985  0600        ld    b,00
F987  09          add   hl,bc
F988  F1          pop   af
F989  4F          ld    c,a
F98A  B7          or    a
F98B  C4F2FF      call  nz,FFF2     ldir
F98E  CDBAFB      call  FBBA
F991  E1          pop   hl
F992  C9          ret

****************************************** BASIC-Befehl MID$
F993  CD37DD      call  DD37        Test auf nachfolgendes Zeichen
F996  28          db    28          '('
F998  CD86D6      call  D686        Variable holen
F99A  CD3CFF      call  FF3C        Typ 'String', sonst 'Type mismatch'
F99D  E5          push  hl
F99E  EB          ex    de,hl
F99F  CD21FB      call  FB21
F9A2  E3          ex    (sp),hl
F9A3  CD37DD      call  DD37        Test auf nachfolgendes Zeichen
F9A6  2C          db    2C          ','
F9A7  CD6DCE      call  CE6D        8-Bit-Wert ungleich Null holen
F9AA  47          ld    b,a
F9AB  CDFBF9      call  F9FB        3. Argument holen (Default = 255)
F9AE  4B          ld    c,e
F9AF  CD37DD      call  DD37        Test auf nachfolgendes Zeichen
F9B2  29          db    29          ')'
F9B3  CD37DD      call  DD37        Test auf nachfolgendes Zeichen
F9B6  EF          db    EF          '='
F9B7  C5          push  bc
```

```
F9B8  CD9FCE     call  CE9F        Stringausdruck und -Parameter holen
F9BB  78         ld    a,b
F9BC  C1         pop   bc
F9BD  E3         ex    (sp),hl
F9BE  0C         inc   c
F9BF  0D         dec   c
F9C0  2825       jr    z,F9E7
F9C2  F5         push  af
F9C3  7E         ld    a,(hl)
F9C4  90         sub   b
F9C5  DA9CFA     jp    c,FA9C      'Improper argument'
F9C8  3C         inc   a
F9C9  B9         cp    c
F9CA  3801       jr    c,F9CD
F9CC  79         ld    a,c
F9CD  4F         ld    c,a
F9CE  78         ld    a,b
F9CF  3D         dec   a
F9D0  23         inc   hl
F9D1  86         add   a,(hl)
F9D2  23         inc   hl
F9D3  66         ld    h,(hl)
F9D4  6F         ld    l,a
F9D5  8C         adc   a,h
F9D6  95         sub   l
F9D7  67         ld    h,a
F9D8  F1         pop   af
F9D9  47         ld    b,a
F9DA  EB         ex    de,hl
F9DB  79         ld    a,c
F9DC  B8         cp    b
F9DD  3801       jr    c,F9E0
F9DF  78         ld    a,b
F9E0  4F         ld    c,a
F9E1  0600       ld    b,00
F9E3  B7         or    a
F9E4  C4F2FF     call  nz,FFF2     ldir
F9E7  E1         pop   hl
F9E8  C9         ret

******************************************* String und 8-Bit-Wert holen
F9E9  CDA5CE     call  CEA5        Stringausdruck holen
F9EC  CD37DD     call  DD37        Test auf nachfolgendes Zeichen
F9EF  2C         db    2C          ','
F9F0  E5         push  hl
F9F1  2AC2B0     ld    hl,(B0C2)
F9F4  E3         ex    (sp),hl
F9F5  CD67CE     call  CE67        8-Bit-Wert holen
F9F8  47         ld    b,a
F9F9  D1         pop   de
F9FA  C9         ret
```

```
****************************************** 3. Argument für MID$ holen
F9FB  1EFF      ld    e,FF        Default 255
F9FD  7E        ld    a,(hl)
F9FE  FE29      cp    29          ')'
FA00  C8        ret   z
FA01  CD37DD    call  DD37        Test auf nachfolgendes Zeichen
FA04  2C        db    2C          ','
FA05  CD67CE    call  CE67        8-Bit-Wert holen
FA08  5F        ld    e,a
FA09  C9        ret

******************************************** BASIC-Funktion LEN
FA0A  CDDAFB    call  FBDA        Stringparameter holen, Länge nach a
FA0D  C30AFF    jp    FF0A        Akkuinhalt als Integerzahl übernehmen

******************************************** BASIC-Funktion ASC
FA10  CD70FA    call  FA70        ASCII-Kode des ersten Zeichens
FA13  C30AFF    jp    FF0A        Akkuinhalt als Integerzahl übernehmen

******************************************* BASIC-Funktion CHR$
FA16  CD92FA    call  FA92        CINT, < 256
FA19  F5        push  af
FA1A  3E01      ld    a,01        Länge 1
FA1C  CD19FC    call  FC19        Platz reservieren, Descriptor ablegen
FA1F  F1        pop   af
FA20  12        ld    (de),a      ASCII-Kode als String ablegen
FA21  C3BAFB    jp    FBBA

*************************************************** INKEY$
FA24  E5        push  hl
FA25  CD2AFA    call  FA2A
FA28  E1        pop   hl
FA29  C9        ret

FA2A  CD39C4    call  C439        KM READ CHAR
FA2D  38EA      jr    c,FA19      Taste gedrückt ?
FA2F  AF        xor   a           nein
FA30  32BAB0    ld    (B0BA),a    Stringdescriptor, Länge
FA33  C3BAFB    jp    FBBA

************************************************** STRING$
FA36  CD67CE    call  CE67        8-Bit-Wert holen, Länge
FA39  4F        ld    c,a
FA3A  CD37DD    call  DD37        Test auf nachfolgendes Zeichen
FA3D  2C        db    2C          ','
FA3E  CDFBCE    call  CEFB        Ausdruck holen
FA41  CD37DD    call  DD37        Test auf nachfolgendes Zeichen
FA44  29        db    29          ')'
FA45  E5        push  hl
FA46  CD45FF    call  FF45        Test auf String
FA49  2805      jr    z,FA50      ja
FA4B  CD92FA    call  FA92        CINT, < 256
FA4E  1803      jr    FA53
```

```
FA50  CD70FA      call  FA70      ASCII-Kode des ersten Zeichens holen
FA53  41          ld    b,c
FA54  4F          ld    c,a
FA55  1807        jr    FA5E

*********************************************** BASIC-Funktion SPACE$
FA57  CD92FA      call  FA92      CINT, < 256
FA5A  47          ld    b,a
FA5B  0E20        ld    c,20'
FA5D  E5          push  hl
FA5E  78          ld    a,b
FA5F  CD19FC      call  FC19      Platz reservieren, Descriptor ablegen
FA62  04          inc   b
FA63  05          dec   b
FA64  2805        jr    z,FA6B
FA66  79          ld    a,c
FA67  12          ld    (de),a
FA68  13          inc   de
FA69  18F8        jr    FA63

FA6B  CDBAFB      call  FBBA
FA6E  E1          pop   hl
FA6F  C9          ret

************************************************** ASCII-Kode holen
FA70  CDDAFB      call  FBDA      Stringparameter holen
FA73  2827        jr    z,FA9C    Leerstring, 'Improper argument'
FA75  1A          ld    a,(de)    Kode des ersten Zeichens
FA76  C9          ret

************************************************ BASIC-Funktion VAL
FA77  CDDAFB      call  FBDA      Stringparameter holen
FA7A  CA0AFF      jp    z,FF0A    Leerstring, dann Null
FA7D  EB          ex    de,hl
FA7E  E5          push  hl
FA7F  5F          ld    e,a
FA80  1600        ld    d,00
FA82  19          add   hl,de
FA83  5E          ld    e,(hl)
FA84  72          ld    (hl),d
FA85  E3          ex    (sp),hl
FA86  D5          push  de
FA87  CDA3EC      call  ECA3
FA8A  D1          pop   de
FA8B  E1          pop   hl
FA8C  73          ld    (hl),e
FA8D  D8          ret   c
FA8E  1E0D        ld    e,0D      'Type mismatch'
FA90  180C        jr    FA9E      Fehlermeldung ausgeben

************************************************** CINT, Test < 256
FA92  E5          push  hl
FA93  CD8DFE      call  FE8D      CINT
FA96  EB          ex    de,hl
FA97  E1          pop   hl
```

```
FA98  7A          ld    a,d          Hi-Byte
FA99  B7          or    a            Null ?
FA9A  7B          ld    a,e          Lo-Byte laden
FA9B  C8          ret   z
FA9C  1E05        ld    e,05         'Improper Argument'
FA9E  C394CA      jp    CA94         Fehlermeldung ausgeben

******************************************* BASIC-Funktion INSTR
FAA1  CDFBCE      call  CEFB         Ausdruck holen
FAA4  CD45FF      call  FF45         Test auf String
FAA7  0E01        ld    c,01         Default Startposition 1
FAA9  280F        jr    z,FABA
FAAB  CD92FA      call  FA92         CINT, < 256
FAAE  B7          or    a
FAAF  CA9CFA      jp    z,FA9C       'Improper argument'
FAB2  4F          ld    c,a
FAB3  CD37DD      call  DD37         Test auf nachfolgendes Zeichen
FAB6  2C          db    2C           ','
FAB7  CDA5CE      call  CEA5         Stringausdruck holen
FABA  CD37DD      call  DD37         Test auf nachfolgendes Zeichen
FABD  2C          db    2C           ','
FABE  E5          push  hl
FABF  2AC2B0      ld    hl,(B0C2)
FAC2  E3          ex    (sp),hl
FAC3  CD9FCE      call  CE9F         Stringausdruck und -Parameter holen
FAC6  CD37DD      call  DD37         Test auf nachfolgendes Zeichen
FAC9  29          db    29           ')'
FACA  E3          ex    (sp),hl
FACB  79          ld    a,c
FACC  CDD4FA      call  FAD4
FACF  CD0AFF      call  FF0A         Akkuinhalt als Integerzahl übernehmen
FAD2  E1          pop   hl
FAD3  C9          ret

FAD4  F5          push  af
FAD5  48          ld    c,b
FAD6  D5          push  de
FAD7  CDE8FB      call  FBE8
FADA  E1          pop   hl
FADB  F1          pop   af
FADC  E5          push  hl
FADD  6F          ld    l,a
FADE  60          ld    h,b
FADF  78          ld    a,b
FAE0  BD          cp    l
FAE1  382D        jr    c,FB10
FAE3  2D          dec   l
FAE4  7D          ld    a,l
FAE5  83          add   a,e
FAE6  5F          ld    e,a
FAE7  8A          adc   a,d
FAE8  93          sub   e
FAE9  57          ld    d,a
FAEA  78          ld    a,b
FAEB  95          sub   l
```

```
FAEC  47            ld     b,a
FAED  79            ld     a,c
FAEE  D601          sub    01
FAF0  7D            ld     a,l
FAF1  3C            inc    a
FAF2  381D          jr     c,FB11
FAF4  E3            ex     (sp),hl
FAF5  C5            push   bc
FAF6  D5            push   de
FAF7  E5            push   hl
FAF8  1A            ld     a,(de)
FAF9  BE            cp     (hl)
FAFA  200D          jr     nz,FB09
FAFC  23            inc    hl
FAFD  0D            dec    c
FAFE  2813          jr     z,FB13
FB00  13            inc    de
FB01  05            dec    b
FB02  20F4          jr     nz,FAF8
FB04  E1            pop    hl
FB05  D1            pop    de
FB06  C1            pop    bc
FB07  1807          jr     FB10

FB09  E1            pop    hl
FB0A  D1            pop    de
FB0B  C1            pop    bc
FB0C  13            inc    de
FB0D  05            dec    b
FB0E  20E5          jr     nz,FAF5
FB10  AF            xor    a
FB11  D1            pop    de
FB12  C9            ret

FB13  E1            pop    hl
FB14  D1            pop    de
FB15  C1            pop    bc
FB16  E1            pop    hl
FB17  7C            ld     a,h
FB18  90            sub    b
FB19  3C            inc    a
FB1A  C9            ret

FB1B  112EFB        ld     de,FB2E
FB1E  C374DA        jp     DA74

FB21  E5            push   hl
FB22  7E            ld     a,(hl)
FB23  23            inc    hl
FB24  4E            ld     c,(hl)
FB25  23            inc    hl
FB26  46            ld     b,(hl)
FB27  EB            ex     de,hl
FB28  B7            or     a
FB29  C42EFB        call   nz,FB2E
```

```
FB2C  E1            pop   hl
FB2D  C9            ret

FB2E  2A8DB0        ld    hl,(B08D)      Beginn der Strings
FB31  CDBEFF        call  FFBE           Vergleich hl <> bc
FB34  3007          jr    nc,FB3D
FB36  2A8FB0        ld    hl,(B08F)      Ende der Strings
FB39  CDBEFF        call  FFBE           Vergleich hl <> bc
FB3C  D0            ret   nc
FB3D  EB            ex    de,hl
FB3E  2B            dec   hl
FB3F  2B            dec   hl
FB40  E5            push  hl
FB41  CD8FFB        call  FB8F
FB44  EB            ex    de,hl
FB45  E1            pop   hl
FB46  C3A6FB        jp    FBA6           Stringdescriptor von (de) nach (hl)

FB49  2AC2B0        ld    hl,(B0C2)
FB4C  11BAB0        ld    de,B0BA        Stringdescriptor
FB4F  CDB8FF        call  FFB8           Vergleich hl <> de
FB52  D8            ret   c
FB53  CD8FFB        call  FB8F
FB56  C3BAFB        jp    FBBA

*****************************************************
FB59  2AC2B0        ld    hl,(B0C2)      Zeiger auf Stringdescriptor
FB5C  E5            push  hl
FB5D  7E            ld    a,(hl)         Stringlänge
FB5E  B7            or    a
FB5F  2826          jr    z,FB87         Leerstring ?
FB61  23            inc   hl
FB62  5E            ld    e,(hl)
FB63  23            inc   hl             Stringlänge nach de
FB64  56            ld    d,(hl)
FB65  2A81AE        ld    hl,(AE81)      Programmstart
FB68  CDB8FF        call  FFB8           Vergleich hl <> de
FB6B  301E          jr    nc,FB8B        String steht vor Programm
FB6D  2A8FB0        ld    hl,(B08F)      Ende der Strings
FB70  CDB8FF        call  FFB8           Vergleich hl <> de
FB73  3816          jr    c,FB8B         String steht oberhalb Stringbereich
FB75  2A83AE        ld    hl,(AE83)      Programmende
FB78  CDB8FF        call  FFB8           Vergleich hl <> de
FB7B  300A          jr    nc,FB87        String steht im Programm
FB7D  E1            pop   hl
FB7E  E5            push  hl
FB7F  119CB0        ld    de,B09C
FB82  CDB8FF        call  FFB8           Vergleich hl <> de
FB85  2004          jr    nz,FB8B
FB87  E1            pop   hl
FB88  C3FFFB        jp    FBFF
```

```
FB8B  E1          pop   hl
FB8C  CDFFFB      call  FBFF
FB8F  7E          ld    a,(hl)
FB90  CD19FC      call  FC19          Platz reservieren, Descriptor ablegen
FB93  D5          push  de
FB94  4E          ld    c,(hl)        Stringlänge nach c
FB95  0600        ld    b,00          Hi-Byte Länge Null
FB97  23          inc   hl
FB98  7E          ld    a,(hl)
FB99  23          inc   hl            Stringadresse nach hl
FB9A  66          ld    h,(hl)
FB9B  6F          ld    l,a
FB9C  78          ld    a,b
FB9D  B1          or    c
FB9E  C4F2FF      call  nz,FFF2       ldir, String übertragen
FBA1  D1          pop   de
FBA2  21BAB0      ld    hl,B0BA       Stringdescriptor
FBA5  C9          ret

*************************************** Stringdescriptor von (de) nach (hl)
FBA6  1A          ld    a,(de)
FBA7  13          inc   de
FBA8  77          ld    (hl),a
FBA9  23          inc   hl
FBAA  1A          ld    a,(de)
FBAB  13          inc   de
FBAC  77          ld    (hl),a
FBAD  23          inc   hl
FBAE  1A          ld    a,(de)
FBAF  13          inc   de
FBB0  77          ld    (hl),a
FBB1  23          inc   hl
FBB2  C9          ret

***************************************** Descriptorstack initialisieren
FBB3  219CB0      ld    hl,B09C
FBB6  229AB0      ld    (B09A),hl     Zeiger in Descriptorstack für Strings
FBB9  C9          ret

************************************************************************
FBBA  3E03        ld    a,03          'String'
FBBC  32C1B0      ld    (B0C1),a      als Variablentyp
FBBF  2A9AB0      ld    hl,(B09A)     Zeiger in Descriptorstack
FBC2  22C2B0      ld    (B0C2),hl
FBC5  11BAB0      ld    de,B0BA       Stringdescriptor
FBC8  CDB8FF      call  FFB8          Vergleich hl <> de
FBCB  1E10        ld    e,10          'String expression too complex'
FBCD  CA94CA      jp    z,CA94        Fehlermeldung ausgeben

FBD0  11BAB0      ld    de,B0BA       Stringdescriptor
FBD3  CDA6FB      call  FBA6          Stringdescriptor von (de) nach (hl)
FBD6  229AB0      ld    (B09A),hl     Zeiger in Descriptorstack
FBD9  C9          ret
```

```
************************************************ Stringparameter holen
FBDA E5         push  hl
FBDB CD3CFF     call  FF3C          Typ 'String', sonst 'Type mismatch'
FBDE 2AC2B0     ld    hl,(B0C2)     Adresse des Stringdescriptors
FBE1 CDE8FB     call  FBE8
FBE4 E1         pop   hl
FBE5 78         ld    a,b           Länge in a und b, Adresse in de
FBE6 B7         or    a
FBE7 C9         ret

FBE8 CDFFFB     call  FBFF
FBEB C0         ret   nz
FBEC D5         push  de
FBED 1B         dec   de
FBEE 2A8DB0     ld    hl,(B08D)     Beginn der Strings
FBF1 CDB8FF     call  FFB8          Vergleich hl <> de
FBF4 2007       jr    nz,FBFD
FBF6 58         ld    e,b
FBF7 1600       ld    d,00
FBF9 19         add   hl,de
FBFA 228DB0     ld    (B08D),hl     Beginn der Strings
FBFD D1         pop   de
FBFE C9         ret

FBFF E5         push  hl
FC00 46         ld    b,(hl)
FC01 23         inc   hl
FC02 7E         ld    a,(hl)
FC03 23         inc   hl
FC04 66         ld    h,(hl)
FC05 6F         ld    l,a
FC06 E3         ex    (sp),hl
FC07 EB         ex    de,hl
FC08 2A9AB0     ld    hl,(B09A)     Zeiger in Descriptorstack
FC0B 2B         dec   hl
FC0C 2B         dec   hl
FC0D 2B         dec   hl
FC0E CDB8FF     call  FFB8          Vergleich hl <> de
FC11 2003       jr    nz,FC16
FC13 229AB0     ld    (B09A),hl     Zeiger in Descriptorstack
FC16 EB         ex    de,hl
FC17 D1         pop   de
FC18 C9         ret

************************************ Platz reservieren, Descriptor ablegen
FC19 F5         push  af
FC1A C5         push  bc
FC1B E5         push  hl
FC1C F5         push  af            Stringlänge
FC1D CDD1F5     call  F5D1          Platz im Stringbereich reservieren
FC20 F1         pop   af
FC21 21BAB0     ld    hl,B0BA       Stringdescriptor
FC24 77         ld    (hl),a        Stringlänge
FC25 23         inc   hl
FC26 73         ld    (hl),e
```

```
FC27  23          inc   hl          Stringadresse
FC28  72          ld    (hl),d
FC29  E1          pop   hl
FC2A  C1          pop   bc
FC2B  F1          pop   af
FC2C  C9          ret

*********************************************** BASIC-Funktion FRE
FC2D  CD45FF      call  FF45        Test auf String
FC30  2006        jr    nz,FC38     nein
FC32  CDDAFB      call  FBDA
FC35  CD3EFC      call  FC3E        Garbage Collection
FC38  CD28F6      call  F628        freien Speicherplatz berechnen
FC3B  C360FE      jp    FE60

*********************************************** Garbage Collection
FC3E  C5          push  bc
FC3F  D5          push  de
FC40  E5          push  hl
FC41  2A8FB0      ld    hl,(B08F)   Ende der Strings
FC44  228DB0      ld    (B08D),hl   Beginn der Strings
FC47  210000      ld    hl,0000
FC4A  22BDB0      ld    (B0BD),hl
FC4D  2A89AE      ld    hl,(AE89)   Arrayende
FC50  22BFB0      ld    (B0BF),hl
FC53  CD7BFC      call  FC7B
FC56  2ABDB0      ld    hl,(B0BD)
FC59  7C          ld    a,h
FC5A  B5          or    l
FC5B  281A        jr    z,FC77
FC5D  56          ld    d,(hl)
FC5E  2B          dec   hl
FC5F  5E          ld    e,(hl)
FC60  E5          push  hl
FC61  2B          dec   hl
FC62  4E          ld    c,(hl)
FC63  0600        ld    b,00
FC65  2A8DB0      ld    hl,(B08D)   Beginn der Strings
FC68  EB          ex    de,hl
FC69  09          add   hl,bc
FC6A  2B          dec   hl
FC6B  CDF5FF      call  FFF5        lddr
FC6E  13          inc   de
FC6F  E1          pop   hl
FC70  73          ld    (hl),e
FC71  23          inc   hl
FC72  72          ld    (hl),d
FC73  1B          dec   de
FC74  EB          ex    de,hl
FC75  18CD        jr    FC44

FC77  E1          pop   hl
FC78  D1          pop   de
FC79  C1          pop   bc
FC7A  C9          ret
```

```
FC7B  219CB0      ld    hl,B09C
FC7E  ED5B9AB0    ld    de,(B09A)    Zeiger in Descriptorstack
FC82  CDB8FF      call  FFB8         Vergleich hl <> de
FC85  280F        jr    z,FC96
FC87  7E          ld    a,(hl)
FC88  23          inc   hl
FC89  4E          ld    c,(hl)
FC8A  23          inc   hl
FC8B  46          ld    b,(hl)
FC8C  E5          push  hl
FC8D  EB          ex    de,hl
FC8E  B7          or    a
FC8F  C49CFC      call  nz,FC9C
FC92  E1          pop   hl
FC93  23          inc   hl
FC94  18E8        jr    FC7E

FC96  119CFC      ld    de,FC9C
FC99  C374DA      jp    DA74

FC9C  2A8DB0      ld    hl,(B08D)    Beginn der Strings
FC9F  CDBEFF      call  FFBE         Vergleich hl <> bc
FCA2  D8          ret   c
FCA3  2ABFB0      ld    hl,(B0BF)
FCA6  CDBEFF      call  FFBE         Vergleich hl <> bc
FCA9  D0          ret   nc
FCAA  EB          ex    de,hl
FCAB  22BDB0      ld    (B0BD),hl
FCAE  ED43BFB0    ld    (B0BF),bc
FCB2  C9          ret

*******************************************************************
FCB3  CD2DFF      call  FF2D         numerisches Ergebnis holen
FCB6  D252BD      jp    nc,BD52      Fließkomma-
FCB9  CDA3BD      call  BDA3         Integer-
FCBC  22C2B0      ld    (B0C2),hl
FCBF  21C3B0      ld    hl,B0C3
FCC2  C9          ret

FCC3  CDC2FE      call  FEC2         UNT
FCC6  21C3B0      ld    hl,B0C3
FCC9  C3A6BD      jp    BDA6

*************************************************** BASIC-Operator '+'
FCCC  CD15FE      call  FE15         Typ der Operanden testen
FCCF  3009        jr    nc,FCDA      Fließkomma ?
FCD1  CDACBD      call  BDAC         Integer-Addition hl := hl + de
FCD4  DA0DFF      jp    c,FF0D       kein Überlauf, Ergebnis in hl übernehmen
FCD7  CD4FFE      call  FE4F         nach Fließkomma wandeln
FCDA  CD58BD      call  BD58         Fließkomma-Addition
FCDD  D8          ret   c            kein Überlauf, ok
FCDE  C3F3CA      jp    CAF3         'Overflow'
```

```
****************************************** BASIC-Operator '-'
FCE1  CD15FE     call  FE15        Typ der Operanden testen
FCE4  3009       jr    nc,FCEF     Fließkomma ?
FCE6  CDB2BD     call  BDB2        Integer-Subtraktion hl := de - hl
FCE9  DA0DFF     jp    c,FF0D      kein Überlauf, Ergebnis in hl übernehmen
FCEC  CD4FFE     call  FE4F        nach Fließkomma wandeln
FCEF  CD5EBD     call  BD5E        Fließkomma-Subtraktion
FCF2  D8         ret   c           kein Überlauf, ok
FCF3  18E9       jr    FCDE        'Overflow'

****************************************** BASIC-Operator '*'
FCF5  CD15FE     call  FE15        Typ der Operanden testen
FCF8  3009       jr    nc,FD03     Fließkomma ?
FCFA  CDB5BD     call  BDB5        Integer-Multiplikation mit Vorzeichen
FCFD  DA0DFF     jp    c,FF0D      kein Überlauf, Ergebnis in hl übernehmen
FD00  CD4FFE     call  FE4F        nach Fließkomma wandeln
FD03  CD61BD     call  BD61        Fließkomma-Multiplikation
FD06  D8         ret   c
FD07  18D5       jr    FCDE        'Overflow'

****************************************** arithmetischer Vergleich
FD09  CD15FE     call  FE15        Typ der Operanden testen
FD0C  DAC4BD     jp    c,BDC4      Integer-Vergleich
FD0F  C36ABD     jp    BD6A        Fließkommavergleich

****************************************** BASIC-Operator '/'
FD12  3AC1B0     ld    a,(B0C1)    Variablentyp
FD15  B1         or    c
FD16  FE02       cp    02
FD18  2005       jr    nz,FD1F
FD1A  CD4FFE     call  FE4F        Integer-Operanden nach Fließkomma
FD1D  1803       jr    FD22

FD1F  CD15FE     call  FE15        Typ der Operanden testen
FD22  EB         ex    de,hl
FD23  D5         push  de
FD24  CD64BD     call  BD64        Fließkommadivision
FD27  D1         pop   de
FD28  F5         push  af
FD29  010500     ld    bc,0005
```

```
FD2C  CDF2FF        call   FFF2        ldir
FD2F  F1            pop    af
FD30  D8            ret    c           ok ?
FD31  CAEACA        jp     z,CAEA      'Division by zero'
FD34  C3F3CA        jp     CAF3        'Overflow'

**************************************** BASIC-Operator 'Backslash'
FD37  CD9AFE        call   FE9A
FD3A  EB            ex     de,hl
FD3B  CDB8BD        call   BDB8        Integer-Division mit Vorzeichen
FD3E  DA0DFF        jp     c,FF0D      Ergebnis in hl übernehmen
FD41  2810          jr     z,FD53      'Division by zero'
FD43  210080        ld     hl,8000
FD46  C360FE        jp     FE60

******************************************** BASIC-Operator 'MOD'
FD49  CD9AFE        call   FE9A
FD4C  EB            ex     de,hl
FD4D  CDBBBD        call   BDBB        MOD-Berechnung
FD50  DA0DFF        jp     c,FF0D      Ergebnis in hl übernehmen
FD53  1E0B          ld     e,0B        'Division by zero'
FD55  C394CA        jp     CA94        Fehlermeldung ausgeben

******************************************** BASIC-Operator 'AND'
FD58  CD9AFE        call   FE9A
FD5B  7B            ld     a,e
FD5C  A5            and    l
FD5D  6F            ld     l,a         hl and de
FD5E  7C            ld     a,h
FD5F  A2            and    d
FD60  C30CFF        jp     FF0C

********************************************* BASIC-Operator 'OR'
FD63  CD9AFE        call   FE9A
FD66  7B            ld     a,e
FD67  B5            or     l
FD68  6F            ld     l,a         hl or de
FD69  7A            ld     a,d
FD6A  B4            or     h
FD6B  18F3          jr     FD60

******************************************** BASIC-Operator 'XOR'
FD6D  CD9AFE        call   FE9A
FD70  7B            ld     a,e
FD71  AD            xor    l
FD72  6F            ld     l,a         hl xor de
FD73  7C            ld     a,h
FD74  AA            xor    d
FD75  18E9          jr     FD60

********************************************* BASIC-Operator NOT
FD77  E5            push   hl
FD78  CD8DFE        call   FE8D        CINT
FD7B  7D            ld     a,l
FD7C  2F            cpl    a           Lo-Byte komplementieren
```

```
FD7D  6F              ld     l,a
FD7E  7C              ld     a,h
FD7F  2F              cpl    a
FD80  CD0CFF          call   FF0C           Hi-Byte komplementieren
FD83  E1              pop    hl
FD84  C9              ret

******************************************** BASIC Funktion ABS
FD85  CDA3FD          call   FDA3           SGN
FD88  F0              ret    p              positives Vorzeichen, fertig

******************************************** Vorzeichen umkehren
FD89  E5              push   hl
FD8A  C5              push   bc
FD8B  CD2DFF          call   FF2D           numerisches Ergebnis holen
FD8E  300D            jr     nc,FD9D        Vorzeichenwechsel Fließkomma
FD90  CDC7BD          call   BDC7           Vorzeichenwechsel Integer
FD93  22C2B0          ld     (B0C2),hl
FD96  D5              push   de
FD97  D460FE          call   nc,FE60
FD9A  D1              pop    de
FD9B  1803            jr     FDA0

**************************************** Vorzeichenwechsel Fließkomma
FD9D  CD6DBD          call   BD6D           Vorzeichenwechsel Fließkomma
FDA0  C1              pop    bc
FDA1  E1              pop    hl
FDA2  C9              ret

*******************************************************
FDA3  CD2DFF          call   FF2D           numerisches Ergebnis holen
FDA6  DACABD          jp     c,BDCA         Integer-SGN
FDA9  C5              push   bc
FDAA  CD70BD          call   BD70           Fließkomma-SGN
FDAD  C1              pop    bc
FDAE  C9              ret

************************************************** Zahl runden
FDAF  E5              push   hl
FDB0  79              ld     a,c
FDB1  CD4BFF          call   FF4B           Variablentyp und -Wert übernehmen
FDB4  D1              pop    de
FDB5  CD2DFF          call   FF2D           numerisches Ergebnis holen
FDB8  78              ld     a,b            Rundungsstellen
FDB9  300B            jr     nc,FDC6        Fließkommawert ?
FDBB  B7              or     a
FDBC  F0              ret    p              Rundung hinter dem Komma? fertig
FDBD  CD6AFE          call   FE6A           Integerwert nach Fließkomma wandeln
FDC0  CDCEFD          call   FDCE           Zahl runden
FDC3  C38DFE          jp     FE8D           CINT

******************************************** Fließkommazahl runden
FDC6  B7              or     a              Rundungsstellen
FDC7  2005            jr     nz,FDCE        ungleich Null, dann runden
FDC9  1149BD          ld     de,BD49        Fließkomma nach Integer wandeln
```

```
FDCC 1826        jr    FDF4

**********************************************************
FDCE D5          push  de
FDCF C5          push  bc
FDD0 78          ld    a,b         Rundungsstellen
FDD1 CD55BD      call  BD55        Fließkommazahl mit 10↑a multiplizieren
FDD4 DC49BD      call  c,BD49      Fließkomma nach Integer wandeln
FDD7 78          ld    a,b
FDD8 C1          pop   bc
FDD9 D1          pop   de
FDDA 3008        jr    nc,FDE4
FDDC CD43BD      call  BD43        Integer nach Fließkomma wandlen
FDDF AF          xor   a           Rundungsstellen invertieren
FDE0 90          sub   b           entspricht Division
FDE1 C355BD      jp    BD55        Fließkommazahl mit 10↑a multiplizieren

FDE4 EB          ex    de,hl
FDE5 C34EFF      jp    FF4E

******************************************** BASIC-Funktion FIX
FDE8 114CBD      ld    de,BD4C     FIX-Funktion
FDEB 1803        jr    FDF0

******************************************** BASIC-Funktion INT
FDED 114FBD      ld    de,BD4F     INT-Funktion
FDF0 CD2DFF      call  FF2D        numerisches Ergebnis holen
FDF3 D8          ret   c           Integer ?
FDF4 CDFBFF      call  FFFB        jp (de)
FDF7 D0          ret   nc
FDF8 3AC1B0      ld    a,(B0C1)    Variablentyp
FDFB CD06FE      call  FE06
FDFE D8          ret   c
FDFF CD1DFF      call  FF1D        Variablentyp nach c, Zeiger nach hl
FE02 78          ld    a,b
FE03 C343BD      jp    BD43        Integer nach Fließkomma wandlen

FE06 79          ld    a,c
FE07 FE03        cp    03          'String' ?
FE09 D0          ret   nc
FE0A 7E          ld    a,(hl)
FE0B 23          inc   hl
FE0C 66          ld    h,(hl)
FE0D 6F          ld    l,a
FE0E CDA9BD      call  BDA9        falls positiv Vorzeichen von b übernehmen
FE11 D0          ret   nc
FE12 C30DFF      jp    FF0D        Ergebnis in hl übernehmen

FE15 79          ld    a,c
FE16 FE03        cp    03          'String' ?
FE18 2832        jr    z,FE4C      ja, 'Type mismatch'
FE1A 3AC1B0      ld    a,(B0C1)    Variablentyp
FE1D FE03        cp    03          'String'
FE1F 282B        jr    z,FE4C      ja, 'Type mismatch'
FE21 B9          cp    c
```

```
FE22  2817      jr    z,FE3B
FE24  300C      jr    nc,FE32
FE26  E5        push  hl
FE27  21C1B0    ld    hl,B0C1      Variablentyp
FE2A  71        ld    (hl),c
FE2B  23        inc   hl
FE2C  CD63FE    call  FE63         Integerzahl nach Fließkomma umwandlen
FE2F  D1        pop   de
FE30  B7        or    a
FE31  C9        ret

FE32  CD63FE    call  FE63         Integerzahl nach Fließkomma umwandlen
FE35  EB        ex    de,hl
FE36  21C2B0    ld    hl,B0C2
FE39  B7        or    a
FE3A  C9        ret

FE3B  EE02      xor   02
FE3D  2805      jr    z,FE44
FE3F  EB        ex    de,hl
FE40  21C2B0    ld    hl,B0C2
FE43  C9        ret

FE44  5E        ld    e,(hl)
FE45  23        inc   hl
FE46  56        ld    d,(hl)
FE47  2AC2B0    ld    hl,(B0C2)
FE4A  37        scf
FE4B  C9        ret

FE4C  C340FF    jp    FF40         'Type mismatch'

************************************** Integer-Operanden nach Fließkomma
FE4F  2AC2B0    ld    hl,(B0C2)    ersten Operand
FE52  CD6AFE    call  FE6A         umwandeln
FE55  2A8BB0    ld    hl,(B08B)    BASIC-Stackpointer, zweiter Operand
FE58  CD63FE    call  FE63         umwandeln
FE5B  EB        ex    de,hl
FE5C  21C2B0    ld    hl,B0C2
FE5F  C9        ret

FE60  AF        xor   a
FE61  1808      jr    FE6B         nach Fließkomma wandeln

************************************** Integerzahl nach Fließkomma umwandlen
FE63  5E        ld    e,(hl)
FE64  23        inc   hl
FE65  56        ld    d,(hl)
FE66  2B        dec   hl
FE67  7A        ld    a,d
FE68  1808      jr    FE72

************************************** Integerzahl nach Fließkomma wandeln
FE6A  7C        ld    a,h
FE6B  EB        ex    de,hl
```

```
FE6C  21C1B0      ld     hl,B0C1          Variablentyp
FE6F  3605        ld     (hl),05          'Real'
FE71  23          inc    hl
FE72  EB          ex     de,hl
FE73  F5          push   af
FE74  B7          or     a
FE75  FCC7BD      call   m,BDC7           falls negativ Integer Vorzeichenwechsel
FE78  F1          pop    af
FE79  C340BD      jp     BD40             Integerzahl nach Fließkomma wandeln

*********************************** 4-Byte-Wert nach Fließkomma wandeln
FE7C  22C2B0      ld     (B0C2),hl        Lo-Word
FE7F  EB          ex     de,hl
FE80  22C4B0      ld     (B0C4),hl        Hi-Word
FE83  21C1B0      ld     hl,B0C1          Variablentyp
FE86  3605        ld     (hl),05          'Real'
FE88  23          inc    hl               Zeiger auf 4-Byte-Wert
FE89  AF          xor    a
FE8A  C343BD      jp     BD43             nach Fließkomma wandeln

********************************************* BASIC-Funktion CINT
FE8D  CD93FE      call   FE93
FE90  D8          ret    c
FE91  183F        jr     FED2             'Overflow'

FE93  CDA5FE      call   FEA5
FE96  22C2B0      ld     (B0C2),hl
FE99  C9          ret

FE9A  79          ld     a,c
FE9B  CDACFE      call   FEAC
FE9E  EB          ex     de,hl
FE9F  DCA5FE      call   c,FEA5
FEA2  D8          ret    c
FEA3  182D        jr     FED2             'Overflow'

FEA5  21C1B0      ld     hl,B0C1          Variablentyp
FEA8  7E          ld     a,(hl)
FEA9  3602        ld     (hl),02          'Integer'
FEAB  23          inc    hl
FEAC  FE03        cp     03               'String' ?
FEAE  380D        jr     c,FEBD
FEB0  CA40FF      jp     z,FF40           'Type mismatch'
FEB3  C5          push   bc
FEB4  CD46BD      call   BD46             Fließkommazahl nach Integer konvertieren
FEB7  47          ld     b,a
FEB8  DCA9BD      call   c,BDA9           Vorzeichen b in Integerzahl hl übernehmen
FEBB  C1          pop    bc
FEBC  C9          ret

********************************************* Integerwert (hl) nach hl
FEBD  7E          ld     a,(hl)
FEBE  23          inc    hl
FEBF  66          ld     h,(hl)
FEC0  6F          ld     l,a
```

```
FEC1  C9             ret

*************************************************** BASIC-Funktion UNT
FEC2  CD2DFF         call   FF2D        numerisches Ergebnis holen
FEC5  D8             ret    c           Integer ?
FEC6  CD46BD         call   BD46        Fließkommazahl nach Integer konvertieren
FEC9  3007           jr     nc,FED2     'Overflow'
FECB  47             ld     b,a
FECC  FCA9BD         call   m,BDA9      Vorzeichen b in Integerzahl hl übernehmen
FECF  DA0DFF         jp     c,FF0D      Integerzahl in hl übernehmen
FED2  1E06           ld     e,06        'Overflow'
FED4  C394CA         jp     CA94        Fehlermeldung ausgeben

FED7  E5             push   hl
FED8  D5             push   de
FED9  C5             push   bc
FEDA  21C1B0         ld     hl,B0C1     Variablentyp
FEDD  BE             cp     (hl)
FEDE  C4E5FE         call   nz,FEE5
FEE1  C1             pop    bc
FEE2  D1             pop    de
FEE3  E1             pop    hl
FEE4  C9             ret

FEE5  D603           sub    03
FEE7  38A4           jr     c,FE8D      CINT
FEE9  CA3CFF         jp     z,FF3C      Typ 'String', sonst 'Type mismatch'
*************************************************** BASIC-Funktion CREAL
FEEC  CD2DFF         call   FF2D        numerisches Ergebnis holen
FEEF  DA6AFE         jp     c,FE6A      Integer, dann umwandeln
FEF2  C9             ret

*********************************************** Fließkommawert auf Null setzen
FEF3  E5             push   hl
FEF4  210000         ld     hl,0000
FEF7  22C2B0         ld     (B0C2),hl
FEFA  22C4B0         ld     (B0C4),hl
FEFD  22C5B0         ld     (B0C5),hl
FF00  E1             pop    hl
FF01  C9             ret

*************************************************** BASIC-Funktion SGN
FF02  CDA3FD         call   FDA3        SGN
FF05  6F             ld     l,a
FF06  87             add    a,a
FF07  9F             sbc    a,a
FF08  1802           jr     FF0C

******************************************* Akkuinhalt als Intgerzahl übernehmen
FF0A  6F             ld     l,a         Lo-Byte
FF0B  AF             xor    a           Hi-Byte löschen
FF0C  67             ld     h,a

*********************************************** Integerzahl in hl übernehmen
FF0D  22C2B0         ld     (B0C2),hl   Zahl nach B0C2
```

```
FF10  3E02      ld    a,02        Typ auf 'Integer'
FF12  32C1B0    ld    (B0C1),a    Variablentyp
FF15  C9        ret

**************************************** Variablentyp auf Fließkomma
FF16  21C2B0    ld    hl,B0C2     Zeiger auf Fließkommazahl
FF19  3E05      ld    a,05        Typ auf 'Real'
FF1B  18F5      jr    FF12

************************************ Variablentyp holen, hl zeigt auf Variable
FF1D  21C1B0    ld    hl,B0C1     Variablentyp
FF20  4E        ld    c,(hl)      nach c
FF21  23        inc   hl          hl zeigt auf Variable
FF22  C9        ret

********************************************** Variablentyp holen
FF23  3AC1B0    ld    a,(B0C1)    Variablentyp
FF26  C9        ret

************************************************* Test auf String
FF27  3AC1B0    ld    a,(B0C1)    Variablentyp
FF2A  FE03      cp    03          'String' ?
FF2C  C9        ret

**************************************** Numerisches Ergebnis holen
FF2D  3AC1B0    ld    a,(B0C1)    Variablentyp
FF30  FE03      cp    03          String ?
FF32  280C      jr    z,FF40      ja, 'Type mismatch'
FF34  2AC2B0    ld    hl,(B0C2)   Integerwert laden
FF37  D8        ret   c           kein Fließkomma, fertig
FF38  21C2B0    ld    hl,B0C2     Adresse der Fließkommazahl
FF3B  C9        ret

*****************************************************************
FF3C  CD45FF    call  FF45        Test auf String
FF3F  C8        ret   z           ja, ok
FF40  1E0D      ld    e,0D        'Type mismatch'
FF42  C394CA    jp    CA94        Fehlermeldung ausgeben

************************************************* Test auf String
FF45  3AC1B0    ld    a,(B0C1)    Variablentyp
FF48  FE03      cp    03          'String' ?
FF4A  C9        ret

*****************************************************************
FF4B  32C1B0    ld    (B0C1),a    Variablentyp
FF4E  11C2B0    ld    de,B0C2
FF51  1813      jr    FF66

************************************** Ergebnis auf BASIC-Stack ablegen
FF53  D5        push  de
FF54  E5        push  hl
FF55  3AC1B0    ld    a,(B0C1)    Variablentyp
FF58  4F        ld    c,a
FF59  CDB0F5    call  F5B0        Platz im BASIC-Stack reservieren
```

```
FF5C  CD62FF         call   FF62            auf Stack ablegen
FF5F  E1             pop    hl
FF60  D1             pop    de
FF61  C9             ret

******************************************** Variable nach (hl) kopieren
FF62  EB             ex     de,hl
FF63  21C2B0         ld     hl,B0C2
FF66  C5             push   bc
FF67  3AC1B0         ld     a,(B0C1)        Variablentyp
FF6A  4F             ld     c,a
FF6B  0600           ld     b,00
FF6D  EDB0           ldir                   Ergebnis kopieren
FF6F  C1             pop    bc
FF70  C9             ret

********************************************** Test auf Buchstaben
FF71  CD8AFF         call   FF8A            Klein- in Großbuchstaben konvertieren
FF74  FE41           cp     41              'A'
FF76  3F             ccf
FF77  D0             ret    nc
FF78  FE5B           cp     5B              'Z'+1
FF7A  C9             ret

************************************** Test auf alphanumerisches Zeichen
FF7B  CD71FF         call   FF71            Test auf Buchstaben
FF7E  D8             ret    c               ja
FF7F  FE2E           cp     2E              '.'
FF81  37             scf
FF82  C8             ret    z
FF83  FE30           cp     30              '0'
FF85  3F             ccf
FF86  D0             ret    nc
FF87  FE3A           cp     3A              '9'+1
FF89  C9             ret

*********************************** Konvertierung Klein- zu Großbuchstaben
FF8A  FE61           cp     61              'a'
FF8C  D8             ret    c
FF8D  FE7B           cp     7B              'z'+1
FF8F  D0             ret    nc
FF90  D620           sub    20              'a'-'A'
FF92  C9             ret

************************************** nachfolgende Tabelle durchsuchen
FF93  F5             push   af
FF94  C5             push   bc
FF95  46             ld     b,(hl)          Tabellenlänge laden
FF96  23             inc    hl
FF97  E5             push   hl              Rücksprungadresse bei negativer Suche
FF98  23             inc    hl              Zeiger auf nächstes Tabellenelement
FF99  23             inc    hl
FF9A  BE             cp     (hl)            Zeichen vergleichen
FF9B  23             inc    hl              Zeiger erhöhen
FF9C  2804           jr     z,FFA2          gefunden
```

```
FF9E  05          dec   b          Zähler erniedrigen
FF9F  20F7        jr    nz,FF98    Tabelle noch nicht zu Ende ?
FFA1  E3          ex    (sp),hl    Rücksprungadresse laden
FFA2  F1          pop   af
FFA3  7E          ld    a,(hl)
FFA4  23          inc   hl
FFA5  66          ld    h,(hl)     Adresse nach hl
FFA6  6F          ld    l,a
FFA7  C1          pop   bc
FFA8  F1          pop   af
FFA9  C9          ret

************************************* Speicherbereich (hl) durchsuchen
FFAA  C5          push  bc         bis (hl) = a (c=1) oder (hl) = 0 (c=0)
FFAB  4F          ld    c,a        a nach c
FFAC  7E          ld    a,(hl)
FFAD  B7          or    a
FFAE  2805        jr    z,FFB5     Null ?
FFB0  23          inc   hl
FFB1  B9          cp    c
FFB2  20F8        jr    nz,FFAC    gleich ürsprunglicher a ?
FFB4  37          scf              Carry setzen
FFB5  79          ld    a,c
FFB6  C1          pop   bc
FFB7  C9          ret

*********************************************** Test hl = de ?
FFB8  7C          ld    a,h
FFB9  92          sub   d          h - d
FFBA  C0          ret   nz
FFBB  7D          ld    a,l
FFBC  93          sub   e          l - e
FFBD  C9          ret

*********************************************** Test hl = bc ?
FFBE  7C          ld    a,h
FFBF  90          sub   b          h - b
FFC0  C0          ret   nz
FFC1  7D          ld    a,l        l - c
FFC2  91          sub   c
FFC3  C9          ret

************************************************* de := de - hl
FFC4  C5          push  bc         bc retten
FFC5  47          ld    b,a        a retten
FFC6  7D          ld    a,l
FFC7  93          sub   e          e - l
FFC8  5F          ld    e,a
FFC9  7C          ld    a,h
FFCA  9A          sbc   a,d        d - h
FFCB  57          ld    d,a
FFCC  78          ld    a,b        a zurück
FFCD  C1          pop   bc         bc zurück
FFCE  C9          ret
```

```
FFCF  C5          push  bc          bc retten
FFD0  47          ld    b,a         a retten
FFD1  7D          ld    a,l
FFD2  93          sub   e           l - e
FFD3  6F          ld    l,a
FFD4  7C          ld    a,h
FFD5  9A          sbc   a,d         h - d
FFD6  67          ld    h,a
FFD7  78          ld    a,b         a zurück
FFD8  C1          pop   bc          bc zurück
FFD9  C9          ret

*********************************************************** bc := hl - de
FFDA  E5          push  hl          hl retten
FFDB  67          ld    h,a         a retten
FFDC  E3          ex    (sp),hl     hl wiederherstellen
FFDD  7D          ld    a,l
FFDE  93          sub   e           l - e
FFDF  4F          ld    c,a         nach c
FFE0  7C          ld    a,h
FFE1  9A          sbc   a,d         h - d
FFE2  47          ld    b,a         nach b
FFE3  E3          ex    (sp),hl
FFE4  7C          ld    a,h         a zurück
FFE5  E1          pop   hl          hl zurück
FFE6  C9          ret
*********************************************************** hl := hl - bc
FFE7  D5          push  de          de retten
FFE8  57          ld    d,a         a retten
FFE9  7D          ld    a,l
FFEA  91          sub   c           l - c
FFEB  6F          ld    l,a
FFEC  7C          ld    a,h
FFED  98          sbc   a,b         h - b
FFEE  67          ld    h,a
FFEF  7A          ld    a,d         a zurück
FFF0  D1          pop   de          de zurück
FFF1  C9          ret

******************************************************** Blocktransfer ldir
FFF2  EDB0        ldir
FFF4  C9          ret

******************************************************** Blocktransfer lddr
FFF5  EDB8        lddr
FFF7  C9          ret

********************************************************* Sprung nach (hl)
FFF8  E9          jp    (hl)

******************************************************** Sprung nach (bc)
FFF9  C5          push  bc
FFFA  C9          ret

******************************************************** Sprung nach (de)
```

```
FFFB  D5        push   de
FFFC  C9        ret

FFFD  C7        rst    0
FFFE  C7        rst    0
FFFF  54
```

4 ANHANG

4.1 Die Betriebssystem Routinen

Wir haben hier die Routinen und Tabellen des Betriebssystems aufgelistet, soweit sie uns bekannt sind.
Achtung: Versuchen Sie nie, die Routinen unter den hier erscheinenden Adressen anzuspringen, wenn Sie nicht mit dem Mechanismus zur Umschaltung der Speicherkonfiguration vertraut sind!
Benutzen Sie lieber die im Kapitel 2.1 aufgeführten Vektoren.
Diese Aufstellung dient in erster Linie dazu, die Vektoren gleichen Namens im Listing schnell auffinden zu können.

0030 RST 6 USER
0040 Bis hierher wird ins Ram kopiert
0044 Restore High Kernel Jumps
005C KL CHOKE OFF
0099 KL TIME PLEASE
00A3 KL TIME SET
00B1 Scan Events
0153 Kick Event
0163 KL NEW FRAME FLY
016A KL ADD FRAME FLY
0176 KL NEW FAST TICKER
017D KL ADD FAST TICKER
0183 Delete Fast Ticker
0189 Ticker Chain bearbeiten
01B3 KL ADD TICKER
01C5 Delete Ticker
01D2 KL INIT EVENT
01E2 KL EVENT
021A KL DO SYNC
0228 KL SYNC RESET
022F Sync Event einhängen
0256 KL NEXT SYNC
0277 KL DONE SYNC
0285 KL DEL SYNCHRONOUS
028E KL DISARM EVENT
0295 KL EVENT DISABLE
029B KL EVENT ENABLE
02A1 KL LOG EXT
02B2 KL FIND COMMAND
0329 KL ROM WALK
0332 KL INIT BACK
0373 Add Event
0382 Delete Event
03B2 KL POLL SYCHRONOUS
03CA RST 7 INTERRUPT ENTRY CONT'D

0401	EXT INTERRUPT ENTRY
040D	KL LOW PCHL CONT'D
0413	RST 1 LOW JUMP CONT'D
0442	KL FAR PCHL CONT'D
044A	KL FAR ICALL CONT'D
0450	RST 3 LOW FAR CALL CONT'D
04A1	KL SIDE PCHL CONT'D
04A7	RST 2 LOW SIDE CALL CONT'D
04BF	RST 5 FIRM JUMP CONT'D
04DB	KL L ROM ENABLE
04E5	KL L ROM DISABLE
04EF	KL U ROM ENABLE
04F9	KL U ROM DISABLE
0503	KL ROM RESTORE
050F	KL ROM SELECT
0514	KL PROBE ROM
051D	KL ROM DESELECT
0533	KL CURR SELECTION
0537	KL LDIR
053D	KL LDDR
0543	Rom off & Konfig. save
055C	RAM LAM
056D	RAM LAM (IX)
0580	RESET CONT'D
05B4	Tabelle 60Hz
05C4	Tabelle 50Hz
05DC	MC BOOT PROGRAM
060B	MC START PROGRAM
065C	Kaltstart
066D	Einschaltmeldung
0693	Copyright–Meldung
06EB	Meldungen ausgeben
06F4	Ladefehler–Meldung
0727	Firmennamen
0776	MC SET MODE
0786	MC CLEAR INKS
0799	MC SET INKS
07AB	Farbe ausgeben
07BA	MC WAIT FLYBACK
07C6	MC SCREEN OFFSET
07E6	MC RESET PRINTER
07F2	MC PRINT CHAR
07F8	MC WAIT PRINTER
0807	MC SEND PRINTER
081B	MC BUSY PRINTER
0826	MC SOUND REGISTER
0846	Scan Keyboard

0888 JUMP RESTORE
08AC Main Jump Adr.
0A28 Basic Jump Adr.
0A8A Move (hl+3)⇒((hl+1)),cnt=(hl)
0AA0 SCR INITIALISE
0AB1 SCR RESET
0ACA SCR SET MODE
0AEC SCR GET MODE
0AF7 SCR MODE CLEAR
0B11 Bit Masken laden
0B2E Bit Masken Mode 2
0B36 Bit Masken Mode 1
0B3A Bit Masken Mode 0
0B3C SCR SET OFFSET
0B45 SCR SET BASE
0B50 SCR GET LOCATION
0B57 SCR CHAR LIMITS
0B64 SCR CHAR POSITION
0BA9 SCR DOT POSITION
0BF9 SCR NEXT BYTE
0C05 SCR PREV BYTE
0C13 SCR NEXT LINE
0C2D SCR PREV LINE
0C49 SCR ACCESS
0C68 SCR WRITE
0C6B SCR PIXELS (FORCE Mode)
0C72 XOR Mode
0C77 AND Mode
0C7D OR Mode
0C82 SCR READ
0C86 SCR INK ENCODE
0CA0 SCR INK DECODE
0CD2 Reset Farben
0CE4 SCR SET FLASHING
0CE8 SCR GET FLASHING
0CEC SCR SET INK
0CF1 SCR SET BORDER
0CF2 Set Colour
0D0A Farbmatrix Eintrag holen
0D14 SCR GET INK
0D19 SCR GET BORDER
0D1A Get Colour
0D2F Ink Adr. holen
0D5B Set Inks on Frame Fly
0D6D Flash Inks
0D81 Params d. lfd Farbsatz holen
0D93 Farbmatrix

0DB3 SCR FILL BOX
0DB7 SCR FLOOD BOX
0DDF SCR CHAR INVERT
0DF2 Farbspeicher adressieren
0DFA SCR HW ROLL
0E3E SCR SW ROLL
0EF3 SCR UNPACK
0F49 SCR REPACK
0FC4 SCR HORIZONTAL
102F SCR VERTICAL
104D Default Farben
1078 TXT INITIALISE
1088 TXT RESET
10A3 Reset Params (alle Fenster)
10E8 TXT STR SELECT
1107 TXT SWAP STREAMS
1122 ldir cnt=15
112A Adr. Fenster Params ⇨ de
113D TXT Default Params setzen
115E TXT SET COLUMN
1169 TXT SET ROW
1174 TXT SET CURSOR
1180 TXT GET CURSOR
118A lfd Fenst. oben,links+hl
1197 lfd Fenster oben,links−hl
11A8 move Cursor
11CE TXT VALIDATE
11DA hl innerhalb Fenstergrenzen?
120C TXT WIN ENABLE
1256 TXT GET WINDOW
1263 TXT DRAW/UNDRAW CURSOR
1268 TXT PLACE/REMOVE CURSOR
1279 TXT CUR ON
1281 TXT CUR OFF
1289 TXT CUR ENABLE
128B Cur Enable Cont'd
129A TXT CUR DISABLE
129C Cur Disable Cont'd
12A9 TXT SET PEN
12AE TXT SET PAPER
12BD TXT GET PEN
12C3 TXT GET PAPER
12C9 TXT INVERSE
12D3 TXT GET MATRIX
12F1 TXT SET MATRIX
12FD TXT SET M TABLE
132A TXT GET M TABLE

1334 TXT WR CHAR
134A TXT WRITE CHAR
137A TXT SET BACK
1387 TXT GET BACK
13A7 TXT SET GRAPHIC
13AB TXT RD CHAR
13C0 TXT UNWRITE
1400 TXT OUTPUT
140C TXT OUT ACTION
144B TXT VDU DISABLE
1451 TXT VDU ENABLE
146B Default Steuerzeichen Sprünge
14CB TXT GET CONTROLS
14D8 07 Klingel
14E3 16 Transparentmode Ein/Aus
14E8 1C =INK Befehl
14F1 1D =BORDER Befehl
14F8 1A Fenster definieren
1504 19 =SYMBOL Befehl
150A 08 CRSR LEFT
150F 09 CRSR RGHT
1514 0A CRSR DOWN
1519 0B CRSR UP
152A 1E CRSR HOME
1530 0D CRSR auf Zeilenanfang
1538 1F =LOCATE Befehl
1540 TXT CLEAR WINDOW
154F 10 Zeichen auf CRS Pos löschen
1556 14 Fenster ab CRS Pos löschen
156D 13 Fenster bis CRS Pos löschen
1584 12 Zeile ab CRS Pos löschen
158E 11 Zeile bis CRS Pos löschen
15B0 GRA INITIALISE
15DF GRA RESET
15F1 GRA MOVE RELATIVE
15FC GRA ASK CURSOR
1612 GRA GET ORIGIN
161A phys Startposition holen
161D phys Zielpos holen + Cur setzen
1657 Add lfd Koord. + rel Koord.
1734 GRA WIN WIDTH
1779 GRA WIN HEIGHT
17A6 GRA GET W WIDTH
17BC GRA GET W HEIGHT
17C5 GRA CLEAR WINDOW
17F6 GRA SET PEN
17FD GRA SET PAPER

1804 GRA GET PEN
180A GRA GET PAPER
1810 GRA PLOT RELATIVE
1813 GRA PLOT ABSOLUTE
1816 GRA PLOT
1824 GRA TEST RELATIVE
1827 GRA TEST ABSOLUTE
182A GRA TEST
1836 GRA LINE RELATIVE
1839 GRA LINE ABSOLUTE
183C GRA LINE
1945 GRA WR CHAR
19E0 KM INITIALISE
1A1E KM RESET
1A3C KM WAIT CHAR
1A42 KM READ CHAR
1A81 KM EXP BUFFER CONT'D
1AB3 Default Exp String
1ABD KM SET EXPAND
1AE5 Exp Buffer aufräumen
1B22 Platz f. neuen Exp String?
1B2E KM GET EXPAND
1B3E Adr. Exp String nach de
1B56 KM WAIT KEY
1B5C KM READ KEY
1BB3 KM GET STATE
1BB7 Update Key State Map
1C2F KM TEST BREAK
1C5C KM GET JOYSTICK
1C69 KM GET DELAY
1C6D KM SET DELAY
1C71 KM ARM BREAK
1C82 KM DISARM BREAK
1C90 KM BREAK EVENT
1CBD KM TEST KEY
1CCD der Key# entspr. Bit holen
1CE5 Bit Masken
1D3E KM GET TRANSLATE
1D43 KM GET SHIFT
1D48 KM GET CONTROL
1D4B Get Key Table
1D52 KM SET TRANSLATE
1D57 KM SET SHIFT
1D5C KM SET CONTROL
1D5F Set Key Table
1D69 Key Translation Table
1DB9 Key SHIFT Table

1E09	Key CTRL Table
1E68	SOUND RESET
1ECB	SOUND HOLD
1EE6	SOUND CONTINUE
1F03	Sound Event
1F61	Scan Sound Queues
1F9F	SOUND QUEUE
204A	SOUND RELEASE
206C	SOUND CHECK
2089	SOUND ARM EVENT
2273	Lautstärke setzen
2338	SOUND AMPL ENVELOPE
233D	SOUND TONE ENVELOPE
2340	Hüllhurve kopieren
2349	SOUND A ADDRESS
234E	SOUND T ADDRESS
2351	Hüllkurve Adresse holen
2370	CAS INITIALISE
237F	CAS SET SPEED
238E	CAS NOISY
2392	CAS IN OPEN
23AB	CAS OUT OPEN
23AF	CAS Open
23FC	CAS IN CLOSE
2401	CAS IN ABANDON
2415	CAS OUT CLOSE
242E	CAS OUT ABANDON
2435	CAS IN CHAR
245B	CAS OUT CHAR
248B	Check Input Buffer Status
248E	Check Buffer Status
2496	CAS TEST EOF
249A	CAS RETURN
24AB	CAS IN DIRECT
24EA	CAS OUT DIRECT
2528	CAS CATALOG
253F	File Header lesen
271F	CAS Meldung (# in b) ausgeben
2780	CAS Meldung (1 Zeichen) ausg.
27C5	Kassetten–Meldungen
2836	CAS READ
283F	CAS WRITE
2851	CAS CHECK
2873	Motor Ein & Keyb. öffnen
29CD	CAS Input RD DATA & Test ESC
2A37	CAS Output WR DATA
2A4B	CAS START MOTOR

2A4F CAS STOP MOTOR
2A51 CAS RESTORE MOTOR
2A98 EDIT
2AC6 EDIT Sprung ausführen
2AE0 EDIT Sprungtabelle 1
2B1C EDIT Sprungtabelle 2
2B2B KLINGEL
2B2F CRSR UP
2B33 CRSR DWN
2B37 CRSR RGHT
2B3B CRSR LEFT
2B42 ESC
2B61 *BREAK*–Meldung
2B69 ENTER
2B75 CRSR RGHT (Puffer)
2B7E CRSR DWN (Puffer)
2B89 CTRL & CRSR RGHT
2B92 CTRL & CRSR DWN
2BAA CRSR LEFT (Puffer)
2BB3 CRSR UP (Puffer)
2BBD CTRL & CRSR LEFT
2BC7 CTRL & CRSR UP
2BF9 CTRL & TAB (Flip Insert)
2C01 Zeichen einfügen
2C3D DEL
2C4A CLR
2C98 SHFT & CRSR RGHT
2C9D SHFT & CRSR LEFT
2CA2 SHFT & CRSR UP
2CA7 SHFT & CRSR DWN
2CEA COPY
2DD9 Zeichen von Keyboard
2DF6 EDIT Sprungadr holen
2E18 FLO Variable von (de) ⇨ (hl) kopieren
2E29 FLO Int ⇨ Flo
2E55 FLO 4–Byte–Wert ⇨ Flo
2E5E FLO 4–Byte–Wert * 256 ⇨ Flo
2E66 FLO Flo ⇨ Int
2E8E FLO Flo ⇨ Int
2EA1 FLO FIX
2EAC FLO INT
2EB6 FLO
2F94 FLO RND Init
2FA1 FLO Set RND Seed
2FB7 FLO RND
2FD1 FLO Zahl mit 10↑a multiplizieren.
2FE6 FLO Letzten RND–Wert holen.

300F FLO LOG10

3014 FLO LOG

3090 FLO EXP

310A FLO SQR

310D FLO Potenzierung

31A3 FLO PI

31AE FLO DEG / RAD

31B2 FLO COS

31BC FLO SIN

3231 FLO TAN

3241 FLO ATN

3337 FLO Subtraktion

333B FLO Subtraktion

333F FLO Addition

3415 FLO Multiplikation

349E FLO Division

3578 FLO Zahl mit 2↑a multiplizieren.

359A FLO Vergleich

35E8 FLO SGN

35F8 FLO Vorzeichenwechsel

3708 INT

370E INT

3715 INT Vorzeichen in b übernehmen.

3728 INT Addition

3730 INT Subtraktion

3731 INT Subtraktion

3739 INT Multiplikation mit Vorzeichen

3750 INT Multiplikation ohne Vorzeichen

377A INT Division mit Vorzeichen

3781 INT MOD

378C INT Division ohne Vorzeichen

37D4 INT Vorzeichenwechsel

37E0 INT SGN

37E9 INT Vergleich

4.2 Referenzen zum System RAM

Im Folgenden finden Sie zu jeder Ram–Adresse, soweit sie im Rom–Listing des Betriebssystems auftaucht, Querverweise auf die Stellen, wo sie benutzt wird.
Das ist dann sehr hilfreich, wenn Sie die Inhalte mit eigenen Programmen manipulieren und plötzlich ein anderer Wert als erwartet darinsteht.
Hier können Sie nun nachsehen, welche Routinen auf die betreffende Adresse zugreifen.

```
B100 : 0066  00F2  011D  0127  061C
B101 : 00EC  061F
B102 : 00F5  00FE  0102
B104 : 00E2  00F8  0114  0132  0142  03E1
B105 : 010A  014E
B187 : 009E  00AC  00B1  010E
B189 : 009A  00A8
B18B : 00A5
B18C : 00BF  016A  0170
B18E : 00C7  017D  0183
B190 : 00DC  0189  01BF  01C5
B192 : 00D2
B193 : 0257  026F  0288  03B9
B194 : 022B  03B2
B195 : 0264  026C  0277  0295  029B  03C3
B196 : 0231  02B2  030A
B1A6 : 02A2  02A6  02BF
B1A8 : 0080  034B  0467  0499  0529  0533
B1A9 : 0060  0086  0096
B1AA : 0348
B1AB : 005D  0083  04B9
B1C8 : 0AEC  0B28
B1C9 : 0B40  0B50  0B84  0BDD  0E24  0E37
B1CA : 0B00
B1CB : 0AA8  0B47  0B53  0B8D  0BE6  0E2C
B1CC : 0C61
B1CD : 0C64
B1CF : 0B20  0BF1  0C8E  0CA2  0F08  0F18  0F32  0F66  0F7D
       0FA1  1015
B1D7 : 0CE4  0CE8  0D8F
B1D8 : 0D88
B1D9 : 0CD5  0D8C
B1EA : 0D32  0D81
B1FB : 0CDE  0D76  0D84
B1FC : 0D06  0D7D
B1FD : 0D5B  0D70
```

B1FE : 0D3C 0D4F
B207 : 0FDC 0FFE
B20C : 10B3 10B7 10EA 1107 1110
B20D : 10A5
B285 : 10A8 1139 1163 116E 117A 1180 11AB 11B1 133F
13B1 1546 1560 1577
B287 : 123E 125D
B288 : 116A 118A 1197 11F3 122D 1256 152A 1543 1559
1570
B289 : 115F 1190 119F 11E1 11E6 1533 1593
B28A : 11FB 1230 1259 1549 155C
B28B : 11DA 11EE 1573 1588
B28C : 1186 11B6
B28D : 1140 1263 1291 12A2
B28E : 1335 1456
B28F : 10CE 10DE 126E 12A9 12BD 12C9 12CF 1391 139F
13C0
B290 : 10C8 11C1 12AE 12C3 13D3 1566 157D 1597
B291 : 1376 1383 1387
B293 : 13A7 140D
B294 : 1320 132A
B295 : 107C
B296 : 1325 1330
B298 : 134E 13C3 13E9
B2B8 : 1415 1447 145C
B2B9 : 142E 143F
B2C3 : 1432 1462 14CB
B328 : 1604 1612 1637
B32A : 1608 1616 164E
B32C : 15F4 15FC 1658
B32E : 15F8 1600 165E
B330 : 1666 16D0 16DA 16DE 1700 1758 17A6 17E2
B332 : 1670 16C7 16E8 16F1 170A 175C 17AA
B334 : 167A 169B 16A4 16BD 1720 178A 17BC 17D9
B336 : 1683 168D 1691 16B3 1716 178E 17C0 17D5
B338 : 17F9 1804 181D 190A 192F
B339 : 17EC 1800 180A 19D8
B33A : 1898 18B3 1911 1936 194A
B33C : 18C6 18D8
B33E : 18C9 18DC
B340 : 18BE 18CD 18E1
B342 : 1841 184E 1859 185D 18A2 18A6 18F7 18FD 1927
193A
B344 : 1845 1860 1864 1872 18A9 18AF 1903 1915 191A
1920
B346 : 18BA 18F1
B43C : 19EF

B446 : 1A24
B4DE : 1A4C 1A6D
B4DF : 1AAF 1ADA
B4E0 : 1A43 1A77
B4E1 : 1A8E 1B44
B4E3 : 1A8A 1B05
B4E5 : 1AAC 1B00 1B11 1B1C 1B22
B4E6 : 1B27
B4E7 : 19EC 1B8D 1BA6 1BB3
B4E8 : 1B76
B4E9 : 1C15 1C69 1C6D
B4EA : 1C4F
B4EB : 1A0F 1BCE 1BFD 1CC5
B4ED : 1BC6 1CBE
B4F1 : 1C5C
B4F3 : 1C2F
B4F4 : 1C62
B4F5 : 1BBA
B4FF : 1BB7 1BCB
B501 : 1BC0
B509 : 1BF1 1C09 1C18
B50A : 1BF6 1C23
B50B : 19E7
B50C : 1C7E 1C84 1C90
B50D : 1C74
B51D : 1E9D 1EEB 1F12 1F48 1FAD 1FD2 2052
B520 : 206F
B522 : 1F74
B539 : 208D
B53C : 1CEE 1CFE 1D26
B53E : 1D0F 1D15
B540 : 1C0D 1D0B 1D22
B541 : 1A01 1D3E 1D52
B543 : 19FD 1D43 1D57
B545 : 19F9 1D48 1D5C
B547 : 19F5 1C02 1CA6 1CAE
B550 : 1F05 20B2
B551 : 1E6D 1EE6 201F 20F5
B552 : 1E6A 1ECB 1F61 2283
B554 : 1F5B 1F97
B555 : 1E70
B55C : 1E80 2125
B59B : 212D 2150
B5DA : 2135 2148
B60A : 219A 2338 2349
B619 : 1E7D 2292
B6FA : 233D 234E

B800 : 238E 2695 2760
B801 : 269A 2705 279F
B802 : 2392 23FC 2401 248B 2528 256E 25A9 27BF
B803 : 24CF 2530 257D
B805 : 2451 2456 24A2 24A6 2580
B807 : 25D6 25E1 25F3
B817 : 258A
B818 : 253F
B819 : 23A6
B81A : 243F 244A 244E 249B 249F 24BC 24D6 259A
B81C : 239E 24B2 24B9 24C1 24D2 256B
B81E : 2594 25CA
B81F : 23A2
B847 : 23AB 2415 242E 245F 24ED 2667
B848 : 2504 251B 262C
B84A : 247F 2484 262F
B84C : 261E 2636
B85C : 265B
B85D : 241F 264D
B85E : 24F9
B85F : 2469 2478 247C 2507 2514 2644 2658
B861 : 2632
B863 : 2624 2660
B864 : 24FC
B866 : 2500
B88C : 254C 25DE 25F6 2692
B89D : 258E
B89F : 2567 2597
B8A3 : 25D0
B8A6 : 24CA
B8CC : 240C 2673
B8CD : 2873 295D 2973
B8CE : 2956 29B3
B8D0 : 2A08 2A1B
B8D1 : 238A
B8D2 : 2A0C
B8D3 : 28B1 2990 29A2 29A6
B8DC : 2C1E 2C35 2C5B 2C67 2DCE
B8DD : 2AA5 2BF9 2BFD 2C04
B8DE : 2C72 2C76 2C83 2C94 2CAC 2CC4 2CD5 2CF0 2CFE
2D11 2D1A 2D36

4.3 Die BASIC-ROM-Routinen

C006 BASIC-Initialisierung
C03F ' BASIC 1.0', LF, LF
C052 BASIC-Befehl EDIT
C064 READY-Modus
C0CC 'Ready', LF
C0D3 AUTO-Modus löschen
C0D6 AUTO-Modus setzen
C0DF BASIC-Befehl AUTO
C12B BASIC-Befehl NEW
C132 BASIC-Befehl CLEAR
C13E Programm und Variablen löschen
C18C Variablen löschen
C1D0 Streamnummer holen
C1E3 Streamnummer testen
C20A BASIC-Befehl PAPER
C212 BASIC-Befehl PEN
C221 BASIC-Befehl BORDER
C22A BASIC-Befehl INK
C23C Argument(e) < 32 holen
C24C Argument < 16 holen
C24F BASIC-Befehl MODE
C25A BASIC-Befehl CLS
C262 BASIC-Funktion VPOS
C276 BASIC-Funktion POS
C290 aktuelle Spaltennummer holen
C2D2 BASIC-Befehl LOCATE
C2E1 BASIC-Befehl WINDOW
C2FD WINDOW SWAP
C312 Argument < 8 holen
C319 BASIC-Befehl TAG
C320 BASIC-Befehl TAGOFF
C327 2 8-Bit-Werte ungleich Null holen
C337 String auf Stream Null ausgeben
C341 String ausgeben
C34E Linefeed ausgeben
C386 Bildschirm initialisieren
C3A8 CR & LF ausgeben
C3B5 Zeichen auf Drucker ausgeben
C3DF aktuelle Druckerposition holen
C3E3 BASIC-BEFEHL WIDTH
C417 reservierte Variable EOF
C424 ein Zeichen vom Eingabestrom holen
C42C auf ein Zeichen von Tastatur warten
C439 Tastatur lesen
C453 Unterbrechung durch 'Break' erlauben

C45E	Break–Event Routine
C46F	Warten auf Tastendruck nach 'ESC'
C48C	BASIC–Befehl ORIGIN
C4C6	BASIC–Befehl DRAW
C4CB	BASIC–Befehl DRAWR
C4D0	BASIC–Befehl PLOT
C4D5	BASIC–Befehl PLOTR
C4E9	BASIC–Funktion TEST
C4EE	BASIC–Funktion TESTR
C505	BASIC–Befehl MOVE
C50A	BASIC–Befehl MOVER
C51A	2 Integerargumente holen
C529	BASIC–Befehl FOR
C5FB	BASIC–Befehl NEXT
C632	offene FOR–Next–Schleife suchen
C6C7	BASIC–Befehl IF
C6E8	BASIC–Befehl GOTO
C6ED	BASIC–Befehl GOSUB
C70F	BASIC–Befehl RETURN
C72E	GOSUB auf BASIC–Stack suchen
C747	BASIC–Befehl WHILE
C776	BASIC–Befehl WEND
C7E3	BASIC–Befehl ON
C807	Event–Verarbeitung
C8CB	BASIC–Befehl ON BREAK
C8E1	BASIC–Befehl DI
C8E7	BASIC–Befehl EI
C8ED	SOUND– und Event–Reset
C940	BASIC–Befehl ON SQ
C95D	Adresse der Sound–Queue berechnen
C971	BASIC–Befehl AFTER
C979	BASIC–Befehl EVERY
C99F	BASIC–Funktion REMAIN
C9B1	Adresse des Event–Blocks berechnen
C9C5	zugehöriges NEXT suchen
CA18	zugehöriges WEND suchen
CA3B	Eingabezeile holen
CA43	Zeile editieren
CA4C	Eingabezeile von Kassette holen
CA84	Fehlernummer und –Zeile löschen
CA8F	BASIC–Befehl ERROR
CA94	Fehlermeldung ausgeben
CB23	'Undefined line'
CB4F	'Break', ' in '
CB5A	BASIC–Befehl STOP
CB65	BASIC–Befehl END
CBC0	BASIC–Befehl CONT

CBE5	ON ERROR
CBF8	ON ERROR GOTO 0
CC03	BASIC–Befehl RESUME
CC45	Zeiger auf Fehlermeldung setzen
CC5B	Fehlermeldungen
CE67	8–Bit–Wert holen
CE6D	8–Bit–Wert ungleich Null holen
CE7C	16–Bit–Wert 0 bis 32767 holen
CE86	Integerwert –32768 bis +32767 holen
CE91	16–Bit–Wert –32768 bis +65535 holen
CE9F	Stringausdruck holen, Parameter bereitstellen
CEA5	Stringausdruck holen
CEB0	Zeilennummernbereich holen
CEE1	Zeilennummer nach de holen
CEFB	Ausdruck holen
CF07	Term holen
CF30	arithmetische Operatoren
CF59	Vergleichsoperatoren
CF81	Tabelle der BASIC–Operatoren
CFA9	arithmetischer Vergleich
CFB9	negatives Vorzeichen
CFC2	BASIC–Operator NOT
CFCE	Ausdruck holen
D00D	Variable holen
D02C	numerische Konstante holen
D070	Ausdruck in Klammern holen
D080	Funktionsberechnung
D0CA	Adressen der reservierten Variablen
D0DC	reservierte Variable ERR
D0E5	reservierte Variable TIME
D0EE	reservierte Variable ERL
D0F4	reservierte Variable HIMEM
D0FA	Variablenpointer, 'Klammeraffe'
D107	reservierte Variable XPOS
D10F	reservierte Variable YPOS
D117	BASIC–Befehl DEF
D130	BASIC–Funktion FN
D190	Adressen der BASIC–Funktionen
D1EA	BASIC–Funktion MIN
D1EE	BASIC–Funktion MAX
D219	BASIC–Funktion ROUND
D246	BASIC–Befehl CAT
D256	BASIC–Befehl OPENOUT
D25F	BASIC–Befehl OPENIN
D298	BASIC–Befehl CLOSEIN
D2A1	BASIC–Befehl CLOSEOUT
D2AD	Kassetten–I/O abbrechen

D2C0	BASIC–Befehl SOUND
D30D	falls vorhanden 8–Bit–Wert holen
D31E	BASIC–Befehl RELEASE
D329	BASIC–Funktion SQ
D341	Argument –128 bis +127 holen
D34E	BASIC–Befehl ENV
D385	BASIC–Befehl ENT
D3FF	Argument 0 bis 4095 holen
D409	BASIC–Funktion INKEY
D423	BASIC–Funktion JOY
D439	BASIC–Befehl KEY
D456	KEY DEF
D494	BASIC–Befehl SPEED
D4AB	SPEED KEY & INK
D4C3	SPEED WRITE
D4DB	reservierte Variable PI
D4E7	BASIC–Befehl DEG
D4EB	BASIC–Befehl RAD
D4EF	BASIC–Funktion SQR
D4F4	BASIC–Operator '↑'
D519	arithmetische Funktion aufrufen
D520	BASIC–Funktion EXP
D525	BASIC–Funktion LOG10
D52A	BASIC–Funktion LOG
D52F	BASIC–Funktion SIN
D534	BASIC–Funktion COS
D539	BASIC–Funktion TAN
D53E	BASIC–Funktion ATN
D53E	'?Random seed'
D559	BASIC–Befehl RANDOMIZE
D584	BASIC–Funktion RND
D5AE	Variablenzeiger rücksetzen
D5D2	Flag für FN löschen
D5EA	Tabellenadresse für Array berechnen
D5FC	Variablentypen A–Z auf 'Real'
D614	BASIC–Befehl DEFSTR
D618	BASIC–Befehl DEFINT
D61C	BASIC–Befehl DEFREAL
D654	BASIC–Befehl LET
D67D	BASIC–Befehl DIM
D686	BASIC–Variable suchen
D690	Variablenadresse holen
D708	Array suchen
D7B5	Variablendimensionierung
D7DB	Test auf Dimensionierte Variable
D906	Variablenname holen
D97F	Variablentyp feststellen

D999	Arraytabelle updaten
D9C0	BASIC–Befehl ERASE
D9CC	Array löschen
DAF8	BASIC–Befehl LINE
DB1A	Eingabe vom aktiven Gerät holen
DB2B	BASIC–Befehl INPUT
DB47	Eingabe holen und umwandeln
DB77	'?Redo from start'
DBAD	Eingabe von Tastatur holen
DCD9	BASIC–Befehl RESTORE
DCEB	BASIC–Befehl READ
DD37	Test auf nachfolgendes Zeichen
DD3F	Blanks überlesen
DD4A	Test auf Ende des Statements
DD55	nächstes Zeichen auf Komma testen
DD61	Blank, TAB und LF überlesen
DD71	Interpreterschleife
DDAB	BASIC–Befehl ausführen
DDD2	aktuelle Zeilenadresse holen
DDD6	aktuelle Zeilenadresse holen und Test auf Direktmodus
DDE2	BASIC–Befehl TRON
DDE6	BASIC–Befehl TROFF
DDEB	TRACE–Routine
DE01	Adressen der BASIC–Befehle
DEE1	Zeichen aus Eingabepuffer holen
DFDC	Tabelle der BASIC–Befehle mit Zeilennummer
E0F7	BASIC–Befehl LIST
E10D	BASIC–Zeilen bc – de listen
E145	Zeichen aus Puffer ausgeben
E155	Bildschirmausgabe
E163	BASIC–Zeile in Puffer listen
E277	Ein–Byte–Zahl ausgeben
E27D	Zwei–Byte–Zahl ausgeben
E288	Zeilennummer ausgeben
E2A3	Binärzahl ausgeben
E2AE	Hexzahl ausgeben
E2C8	Fließkommazahl ausgeben
E354	Adressen der BASIC–Befehlsworte
E388	Tabelle der BASIC–Befehlsworte
E64B	Tabelle der BASIC–Operatoren
E676	Programmzeiger löschen
E69D	Zeilenadresse durch Zeilennummer ersetzen
E6BC	Eingabezeile in Interpreterkode wandeln
E6D2	Statement in Interpreterkode wandeln
E728	BASIC–Befehl DELETE
E767	Zeilenadresse holen
E79A	BASIC–Zeile suchen

E7DF	BASIC–Befehl RENUM
E8C1	Test auf indizierte Variable
E8EF	BASIC–Befehl DATA
E8F3	BASIC–Befehle ELSE, REM und '
E9BD	BASIC–Befehl RUN
E9F6	BASIC–Befehl LOAD
EA3C	BASIC–Befehl CHAIN
EAA6	BASIC–Befehl MERGE
EC09	BASIC–Befehl SAVE
EC3D	SAVE ,P
EC5C	SAVE ,B
EC87	SAVE ,A
EE61	ASCII–Ziffer nach binär wandeln
EE79	Integerzahl hl ausgeben
EE82	Integerzahl nach ASCII wandeln
EE9D	Zahl nach ASCII wandeln
EE9F	Zahl formatieren
F114	Umwandlung nach binär
F119	Umwandlung nach hex
F158	BASIC–Funktion PEEK
F15F	BASIC–Befehl POKE
F16D	BASIC–Funktion INP
F177	BASIC–Befehl OUT
F17D	BASIC–Befehl WAIT
F194	16–Bit–Wert und 8–Bit–Wert holen
F1A0	BASIC–Befehlserweiterung suchen
F1BA	BASIC–Befehl CALL
F1F2	TAB–Stops initialisieren
F1F6	BASIC–Befehl ZONE
F1FD	BASIC–Befehl PRINT
F25C	PRINT ,
F277	PRINT SPC
F280	PRINT TAB
F2A0	Integerwert in Klammern holen
F2C4	PRINT USING
F3BA	auf Formatierungszeichen prüfen
F47B	BASIC–Befehl WRITE
F4C4	Speicher konfigurieren
F4EF	BASIC–Befehl MEMORY
F501	Platz für zu ladendes Programm schaffen
F51D	Länge des Stringbereichs berechnen
F52C	Programm– und Variablenzeiger um bc erhöhen
F58E	BASIC–Stack initialisieren
F5A0	Platz im BASIC–Stack freigeben
F5B0	Platz im BASIC–Stack reservieren
F5D1	Platz für String reservieren
F5E6	Test auf Platz im Stringbereich

F5F8 Platz im Variablenbereich reservieren
F618 Test auf Platz im Variablenbereich
F628 freien Speicherplatz berechnen
F69D BASIC–Befehl SYMBOL
F6CD SYMBOL AFTER
F7CB String lesen
F828 String ausgeben
F834 BASIC–Funktion LOWER$
F839 Umwandlung Groß– in Kleinbuchstaben
F842 BASIC–Funktion UPPER$
F863 Stringaddition
F897 Stringvergleich
F8BA BASIC–Funktion BIN$
F8C4 BASIC–Funktion HEX$
F8CE Argumente für BIN$ und HEX$ holen
F8EA BASIC–Funktion DEC$
F91E BASIC–Funktion STR$
F93C BASIC–Funktion LEFT$
F943 BASIC–Funktion RIGHT$
F94B BASIC–Funktion MID$
F993 BASIC–Befehl MID$
F9E9 String und 8–Bit–Wert holen
F9FB 3. Argument für MID$ holen
FA0A BASIC–Funktion LEN
FA10 BASIC–Funktion ASC
FA16 BASIC–Funktion CHR$
FA24 reservierte Variable INKEY$
FA36 BASIC–Funktion STRING$
FA57 BASIC–Funktion SPACE$
FA70 ASCII–Kode holen
FA77 BASIC–Funktion VAL
FA92 Umwandlung nach Integer und Test < 256
FAA1 BASIC–Funktion INSTR
FBB3 Descriptorstack initialisieren
FBDA Stringparameter holen
FC19 Platz reservieren, Descriptor ablegen
FC2D BASIC–Funktion FRE
FC3E Garbage Collection
FCCC BASIC–Operator '+'
FCE1 BASIC–Operator '–'
FCF5 BASIC–Operator '*'
FD09 arithmtischer Vergleich
FD12 BASIC–Operator '/'
FD37 BASIC–Operator 'Backslash'
FD49 BASIC–Operator MOD
FD58 BASIC–Operator AND
FD63 BASIC–Operator OR

FD6D BASIC–Operator XOR
FD85 BASIC–Funktion ABS
FD89 Vorzeichen wechseln
FDE8 BASIC–Funktion FIX
FDED BASIC–Funktion INT
FE4F Integeroperanden nach Fließkomma wandeln
FE6A Integerzahl nach Fließkomma wandeln
FE8D BASIC–Funktion CINT
FEC2 BASIC–Funktion UNT
FEEC BASIC–Funktion CREAL
FF02 BASIC–Funktion SGN
FF0A Akkuinhalt als Integerzahl übernehmen
FF0D Integerzahl in hl übernehmen
FF16 Variablentyp auf Fließkomma setzen
FF23 Variablentyp holen
FF27 Test auf String
FF2D numerisches Ergebnis holen
FF3C Test auf String, sonst 'Type mismatch'
FF45 Test auf String
FF53 Ergebnis auf BASIC–Stack ablegen
FF62 Variable nach (hl) kopieren
FF71 Test auf Buchstaben
FF8A Klein– in Großbuchstaben umwandeln
FF93 Tabelle durchsuchen
FFAA Tabelle durchsuchen
FFB8 Vergleich hl <> de
FFBE Vergleich hl <> bc
FFC4 de := de – hl
FFCF hl := hl – de
FFDA bc := hl – de
FFE7 hl := hl – bc
FFF2 Blocktransfer ldir
FFF5 Blocktransfer lddr
FFF8 jp (hl)
FFF9 jp (bc)
FFFB jp (de)

4.4 Die BASIC-Tokens

Token	Bedeutung
00	Zeilenende
01	':', Ende des Statements
02	Integervariable '%'
03	Stringvariable '$'
04	Realvariable '!'
0D	Variable ohne Kennzeichen
0E	Konstante 0
0F	Konstante 1
10	Konstante 2
11	Konstante 3
12	Konstante 4
13	Konstante 5
14	Konstante 6
15	Konstante 7
16	Konstante 8
17	Konstante 9
19	Ein−Byte−Wert
1A	Zwei−Byte−Wert, dezimal
1B	Zwei−Byte−Wert, binär
1C	Zwei−Byte−Wert, hex
1D	Zeilenadresse
1E	Zeilennummer
1F	Fließkommawert
80	AFTER
81	AUTO
82	BORDER
83	CALL
84	CAT
85	CHAIN
86	CLEAR
87	CLG
88	CLOSEIN
89	CLOSEOUT
8A	CLS
8B	CONT
8C	DATA
8D	DEF
8E	DEFINT
8F	DEFREAL
90	DEFSTR
91	DEG
92	DELETE
93	DIM
94	DRAW
95	DRAWR
96	EDIT
97	ELSE
98	END
99	ENT
9A	ENV
9B	ERASE
9C	ERROR
9D	EVERY
9E	FOR
9F	GOSUB
A0	GOTO
A1	IF
A2	INK
A3	INPUT
A4	KEY
A5	LET
A6	LINE
A7	LIST
A8	LOAD
A9	LOCATE
AA	MEMORY
AB	MERGE
AC	MID$
AD	MODE
AE	MOVE
AF	MOVER
B0	NEXT
B1	NEW
B2	ON
B3	ON BREAK
B4	ON ERROR GOTO 0
B5	ON SQ
B6	OPENIN
B7	OPENOUT
B8	ORIGIN
B9	OUT
BA	PAPER
BB	PEN
BC	PLOT
BD	PLOTR
BE	POKE
BF	PRINT
C0	'
C1	RAD
C2	RANDOMIZE
C3	READ
C4	RELEASE
C5	REM
C6	RENUM

C7	RESTORE
C8	RESUME
C9	RETURN
CA	RUN
CB	SAVE
CC	SOUND
CD	SPEED
CF	SYMBOL
D0	TAG
D1	TAGOFF
D2	TRON
D3	TROFF
D4	WAIT
D5	WEND
D6	WHILE
D7	WIDTH
D8	WINDOW
D9	ZONE
DA	WRITE
DB	DI
DC	EI
E3	ERL
E4	FN
E5	SPC
E6	STEP
E7	SWAP
EA	TAB
EB	THEN
EC	TO
ED	USING
EE	>
EF	=
F0	>=
F1	<
F2	<>
F3	<=
F4	+
F5	−
F6	*
F7	/
F8	↑
F9	'Backslash'
FA	AND
FB	MOD
FC	OR
FD	XOR
FE	NOT
FF	Funktion
CE	STOP

Das Token &FF steht vor einer Funktion. Danach können die nachstehenden Token folgen:

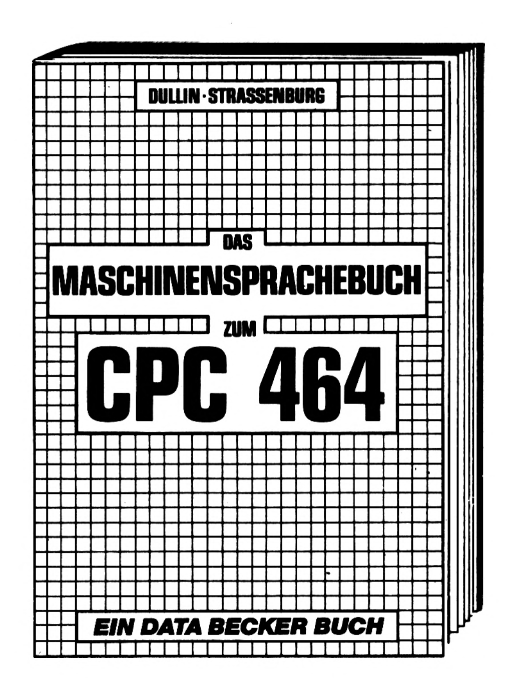
00	ABS
01	ASC
02	ATN
03	CHR$
04	CINT
05	COS
06	CREAL
07	EXP
08	FIX
09	FRE
0A	INKEY
0B	INP
0C	INT
0D	JOY
0E	LEN
0F	LOG
10	LOG10
11	LOWER$
12	PEEK
13	REMAIN
14	SGN
15	SIN
16	SPACE$
17	SQ
18	SQR
19	STR$
1A	TAN
1B	UNT
1C	UPPER$
1D	VAL
40	EOF
41	ERR
42	HIMEM
43	INKEY$
44	PI
45	RND
46	TIME
47	XPOS
48	YPOS
71	BIN$
72	DEC$
73	HEX$
74	INSTR
75	LEFT$
76	MAX
77	MIN
78	POS
79	RIGHT$
7A	ROUND
7B	STRING$
7C	TEST
7D	TESTR
7E	'Improper argument'
7F	VPOS



Deutschlands meistverkaufte Textverarbeitung jetzt in einer speziellen Version für den CPC 464. Erweitert um 80-Zeichen-Darstellung, Tabulatoren, Word Wrap und Trennvorschläge. Natürlich mit deutschem Zeichensatz. Komplett in Maschinensprache und damit superschnell. Durch Menuesteuerung leicht zu bedienen. Läßt sich ideal mit DATAMAT kombinieren. **TEXTOMAT für den CPC 464 kostet einschließlich umfangreichem Handbuch DM 148,–*.**

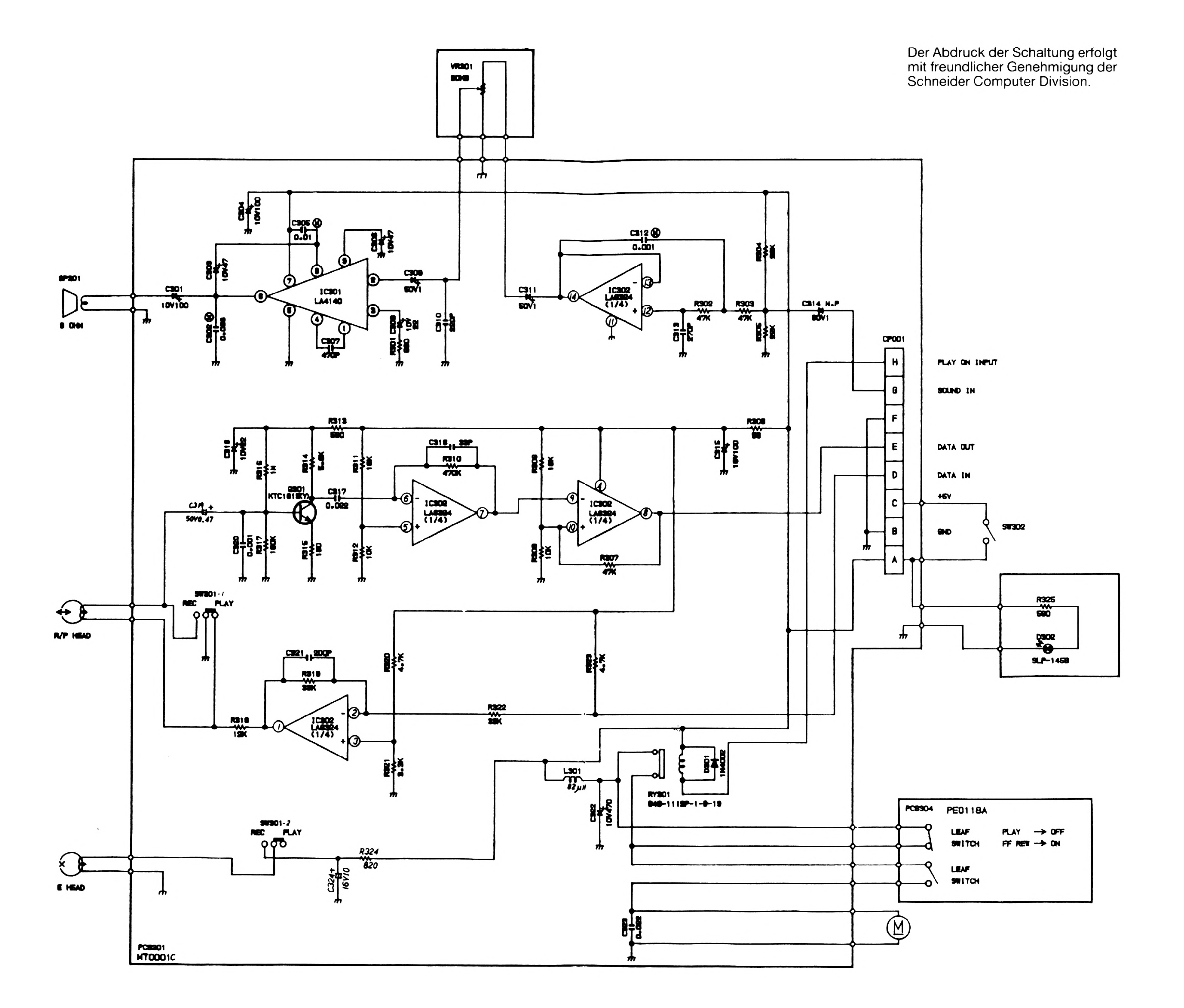
* Unverbindliche Preisempfehlung


DATAMAT
DATA BECKER

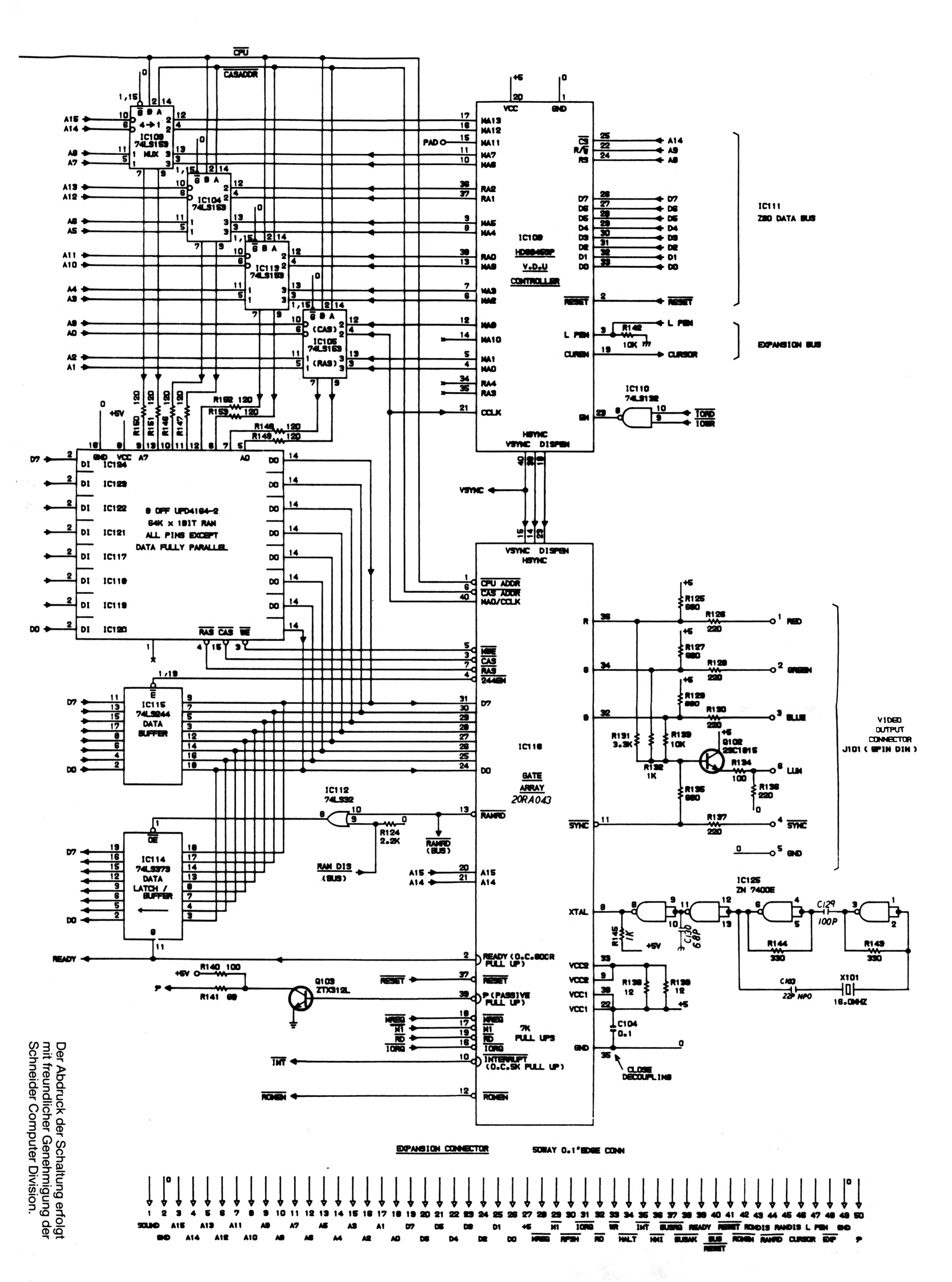


Das Maschinensprachebuch zum CPC 464 für jeden, dem das BASIC nicht mehr ausreicht. Von den Grundlagen der Maschinenspracheprogrammierung über die Arbeitsweise des Z 80-Prozessors und der Beschreibung seiner Befehle bis zur Benutzung von Systemroutinen ist alles ausführlich erklärt. Dazu Assembler, Disassembler und Monitor als Anwenderprogramme. Einstieg in Maschinensprache leicht gemacht!

Das Maschinensprachebuch zum CPC 464, über 300 Seiten, DM 39,–.

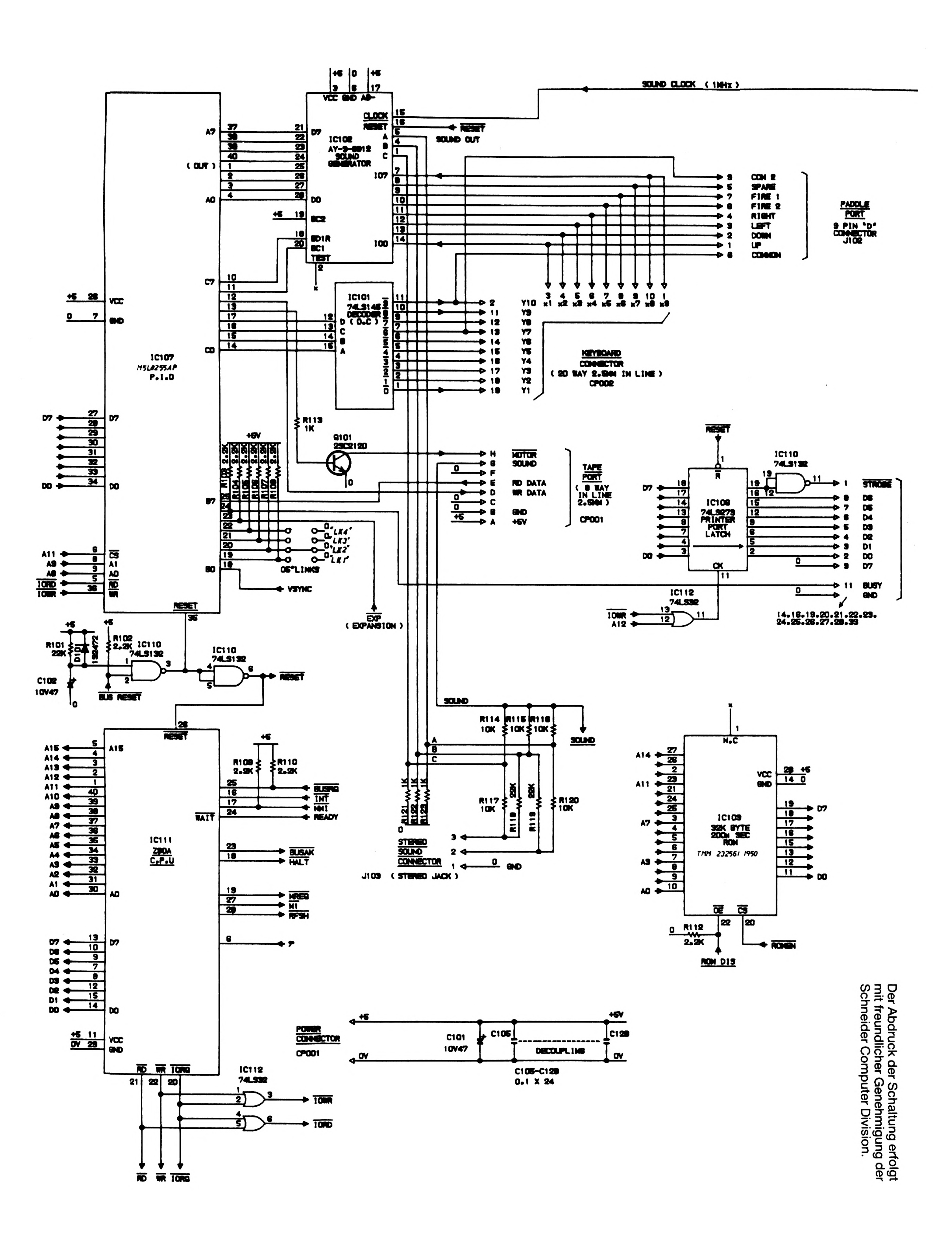


Der Abdruck der Schaltung erfolgt mit freundlicher Genehmigung der Schneider Computer Division.



Der Abdruck der Schaltung erfolgt mit freundlicher Genehmigung der Schneider Computer Division.





Der Abdruck der Schaltung erfolgt mit freundlicher Genehmigung der Schneider Computer Division.



DAS STEHT DRIN:

CPC 464 INTERN ist eine unentbehrliche Hilfe für den fortgeschrittenen Basic-Programmierer und ein absolutes Muß für den Assembler-Programmierer. Hier wurde in monatelanger Kleinarbeit ein Standardwerk geschaffen, das wirklich alle Geheimnisse des CPC 464 enthüllt.

Aus dem Inhalt:

- Das sollten Sie von Ihrem Gerät wissen
- Die Speicheraufteilung
- Der Prozessor
- Besonderheiten des Z 80 im CPC 464
- Das Gate Array
- Der Video-Controller
- Das Video-Ram
- Der Soundchip
- Die Schnittstellen
- Das Betriebssystem
  Nutzung der Routinen am Beispiel Hardcopy
  Der Character-Generator
- Der BASIC-Interpreter
- BASIC und Maschinensprache
- Das BASIC-ROM-Listing
  und vieles mehr

UND GESCHRIEBEN HAT DIESES BUCH:

Das bewährte DATA-BECKER-Autorenteam mit Rolf Brückmann, Lothar Englisch und Klaus Gerits. Alle sind nicht nur begeisterte Programmierer, die Ihren CPC 464 in und auswendig kennen, sondern auch bekannte Autoren vieler weiterer Bücher.

ISBN 3-89011-080-0

Brückmann · Englisch · Gerits / CPC 464 Intern

CPC 464


AMSTRAD
CPC
OCR
MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA
ACME
https://acpc.me/